

**AQUARIUS**<sup>TM</sup>  
HOME COMPUTER SYSTEM

# LOGO

CARTRIDGE INSTRUCTIONS

FOR COLOR TV VIEWING ONLY

MATTEL ELECTRONICS®



---

## **KEY POINTS FOR TROUBLE FREE USE OF YOUR AQUARIUS™ HOME COMPUTER SYSTEM**

- 1.** Place AQUARIUS™ on a flat, sturdy surface, so that air vents on the bottom can do their job. Do NOT place ON CARPETED SURFACES.
- 2.** Keep fingers out of the open end of program cartridges and cartridge port(s).
- 3.** Never lift AQUARIUS™ by any external wires, or by the Mini Expander, or by its hand controllers.
- 4.** Protect both program cartridges and AQUARIUS™ from excessive heat.
- 5.** Always insert cartridge port dust cover when port is not in use.
- 6.** Turn AQUARIUS™ OFF and unplug transformer from the wall outlet when not in use.

## **NOTICE — PROJECTION TV OWNERS**

Some stationary game patterns produced by this product may be permanently imprinted on Projection TV tubes by extended use at high brightness levels. Consult Projection TV Owner's manual before use of this product.

## **WARNING:**

This equipment has been certified to comply with the limits for a Class B computing device pursuant to Subpart J of Part 15 of FCC rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

The system software in the AQUARIUS™ Computer is retained in a Read Only Memory (ROM) device. All portions of this system's software, whether in ROM format or the ROM circuitry are copyrighted and are proprietary information. Any use, reproduction, or publication of any portion of this material without prior written authorization by Microsoft Inc. and Mattel Inc. is strictly prohibited.

## **IMPORTANT**

Mattel Electronics does not assume any liability or responsibility for loss or damage, direct or indirect, caused by or alleged to be caused by any software programs (whether sold by Mattel Electronics or otherwise) or the use made of any such programs by the consumer.

# TABLE OF CONTENTS

## INTRODUCTION 4

A Quick Glance At the Logo Language

## CHAPTER 1:

### "GETTING YOUR FEET WET" 6

Loading and Unloading the  
Logo Cartridge  
Cold Start/Warm Start  
Crashes  
Keyboard and Overlay  
Graphic Screen  
Command Window  
Modes  
Space, Shift, Control (CTL), Return (RTN),  
Reset (RST) Keys  
Special Symbols  
Error Messages

## CHAPTER 2:

### "GETTING STARTED WITH TURTLE GRAPHICS" 13

Take The Turtle For a Walk 14

Forward (FD), Back (BK), Right  
Turn (RT), Left Turn (LT)  
Commands 14

Clear the Screen 16

Graphics Screen 16

Color The Turtle/Background 19

PAPER and INK Command 19

Erase the Turtle Track 21

TILE Command 21

Saving Typing Time Using The

Repeat Command (REP) 22

Chapter 2 Summary 24

## CHAPTER 3:

### "MORE ABOUT TURTLE GRAPHICS" 25

Penstates 25

PEN and Turtle Commands 25

Turtle Shortcuts 27

Directional Commands: East (E),

West (W), North (N), South (S) 27

Positional Command (POS) 30

Heading Command (HD) 31

Changing The Turtle Track 32

TILE Command and Turtle Prints

(TP) Command 37

Chapter 3 Summary

---

## CHAPTER 4:

### "CREATING A TURTLE VOCABULARY"

	39
Creating A New Turtle Word Using Procedures	40
Define Mode	40
Passing Procedures	43
Errors In Typing Procedures	44
Edit Mode	44
Delete	
Change	
Insert	
Chapter 4 Summary	48

## CHAPTER 5:

### "EXPANDING YOUR TURTLE VOCABULARY USING VARIABLES"

	49
Introducing The Variables Symbols —	
Dots (.)	49
Reviewing Variables	51
Using Variables In A Procedure —	
DOT Command	54
Multiple Variable Procedures —	
Back To The House	56
Chapter 5 Summary	60

## CHAPTER 6:

### "ADD LOOPS TO YOUR PROCEDURES"

	61
Looping With Logo — Recursion	61
Conditional Branching With Logo	
IFTRUE/IFFALSE Commands	63
Conditional Statements	64
MAKE and PRINT Command	68
Chapter 6 Summary	76

## CHAPTER 7:

### "MANAGING YOUR AQUARIUS WORKSPACE AND STORAGE"

	77
NO ROOM Message	77
Reorganize The Workspace Using	
ERASE and NAMES Commands	77
Aquarius Mini Expander	78
Saving Logo Programs On The	
Aquarius Data Recorder	79
SAVE Command	
Using CHECK —	
The Check Command	81
Using LOAD — The LOAD Command	82
Using The Aquarius Line Printer,	
The LP Command	83
Chapter 7 Summary	84

---

---

## CHAPTER 8:

### **"MULTIPLE TURTLES AND ANIMATION"** 85

- Creating Multiple Turtles —  
HATCH and ASK Commands 85
- Turtle Shapes and Animation: A Bouncing  
Ball — SHAPE Command 88
- Making An Animated Figure With  
Two Turtles 92
- Review Multiple Turtles and Animation 94
- Chapter 8 Summary 95

## CHAPTER 9:

### **"BEYOND TURTLE GRAPHICS: NUMBER OPERATIONS"** 97

- Numeric Operations in Aquarius Logo 98
- Addition, Subtraction, Multiplication,  
Division, Exponentiation, Integer  
and Random

## CHAPTER 10:

### **"BEYOND TURTLE GRAPHICS: LIST PROCESSING"** 101

- What Is List Processing? 101
- Character Strings, Words, and Lists.  
FIRST, But First (BF), LAST,  
But Last (BL) Commands 101
- Punctuation Marks for List Processing 105

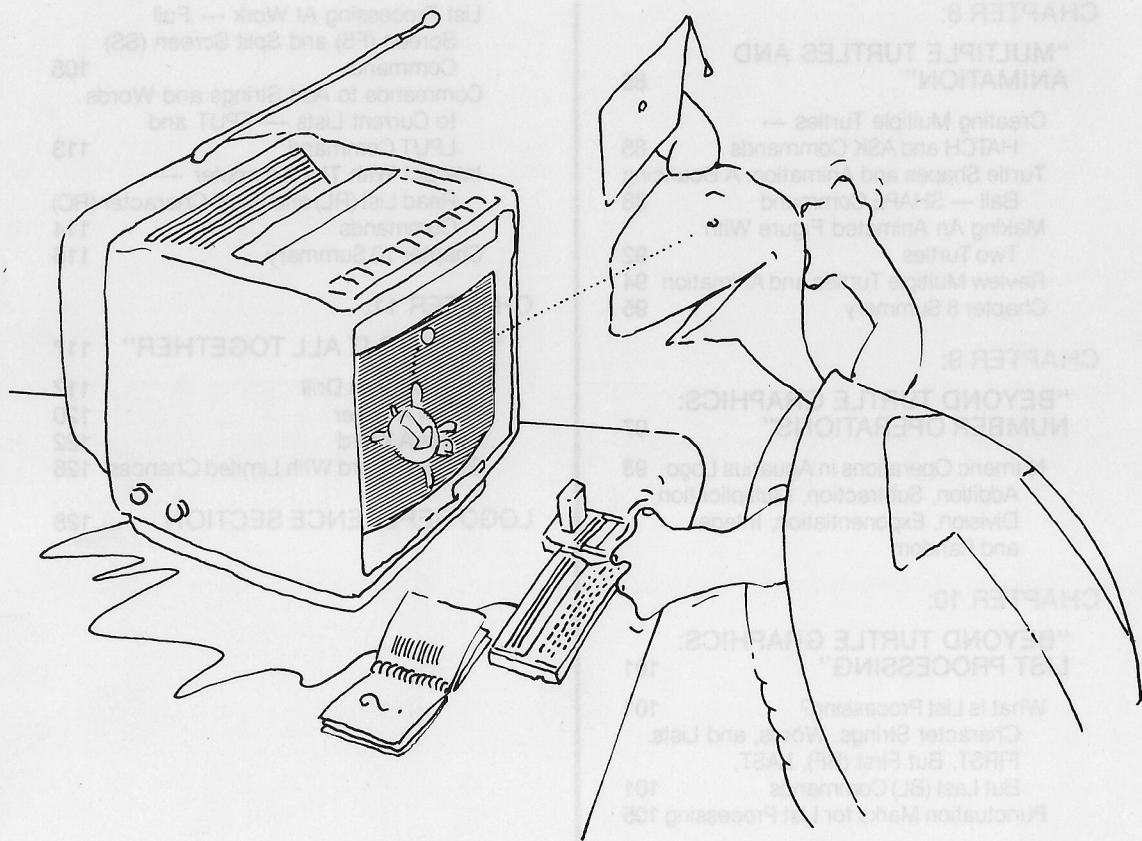
- List Processing At Work — Full  
Screen (FS) and Split Screen (SS)  
Command 108
- Commands to Add Strings and Words  
to Current Lists — FPUT and  
LPUT Command 113
- Interact With The Computer —  
Read List (RL) and Read Character (RC)  
Commands 114
- Chapter 10 Summary 116

## CHAPTER 11:

### **"PUTTING IT ALL TOGETHER"** 117

- Multiplication Drill 117
- Guess A Letter 120
- Guess A Word 122
- Guess A Word With Limited Chances 126

## LOGO REFERENCE SECTION 128



# INTRODUCTION

LOGO is a simple language that lets you communicate with a computer. It's the perfect introduction to computer programming for children and adults! Aquarius LOGO is easy to learn and becomes more complex whenever you want!

The LOGO language has a vocabulary of about 80 words and a few symbols. These are called "primitive commands" and are defined in Appendix B of this book.

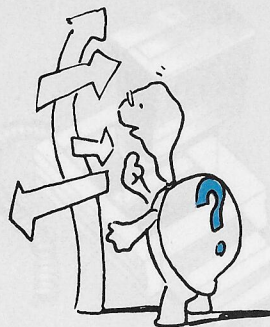
The main feature of Aquarius LOGO is learning to program using turtle graphics! There's an arrow on the screen which is affectionately called a "turtle"! You type in simple instructions. The turtle follows these instructions and immediately creates graphics on the screen.

LOGO also computes arithmetic operations. Keep it simple with addition, subtraction, multiplication and division...or try exponentiation and square roots! The random number function comes in handy when you want to create math games and math drills.

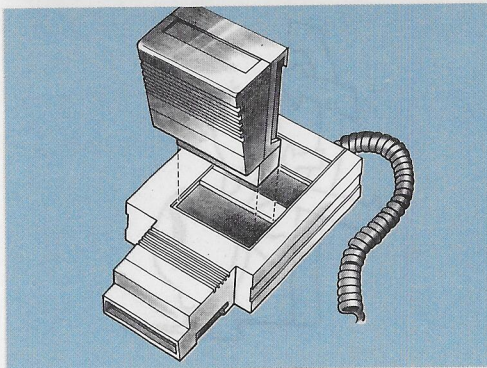
## CHAPTER 1

The third feature of Aquarius LOGO is "list processing". With simple instructions, the computer will edit and manipulate data you have entered. For instance, type in all the birthdays of your friends, then ask the computer to tell you who's birthday falls in the month of May.

We encourage you to read through these instructions and try all the examples. Have fun learning LOGO!



## CHAPTER 1



### TO LOAD YOUR LOGO

**1.** Make sure your Aquarius computer and TV set are plugged into a 110/120 volt AC wall outlet.

**2.** Make sure the computer and TV set are both turned off.

**3.** If you're not using a memory cartridge, insert the LOGO cartridge into the cartridge port of the Aquarius computer or into the front port of your Mini Expander.

If you plan to use a memory cartridge:

**A** Insert the memory cartridge into the rear port of the Mini Expander.

**B** Insert the LOGO cartridge into the front port of the Mini Expander.

**C** Insert the Mini Expander into the cartridge port of the Aquarius computer.

**4.** Connect the Aquarius computer to the TV set.

**5.** Set the Aquarius to channel 3 or 4, whichever is weaker in your area. Then tune your TV set to the same channel.

**6.** Turn on the Aquarius computer; then turn on the TV.

**7.** If you plan to use a cassette recorder or printer, you may plug it in while the Aquarius is running.

8. Press **RST**. The Turtle appears at the center of the screen.
9. The question mark (?) prompt at the bottom of the screen tells you that LOGO is ready for you to use.

## TO REMOVE YOUR LOGO CARTRIDGE

1. Turn off the TV.
2. Turn off the Aquarius computer.
3. If you're not using a Mini Expander, remove the LOGO cartridge from the cartridge port.  
If you are using a Mini Expander, remove the LOGO cartridge from the program port or simply remove the Mini Expander from the Aquarius cartridge port.
4. If you don't plan to use the Aquarius computer for a while, unplug the computer and the peripherals from the 110/120 volt AC wall outlet.

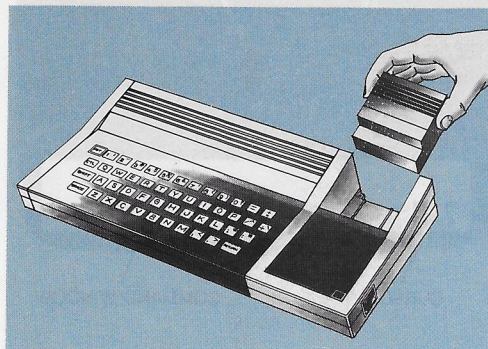
## A REMINDER ABOUT COLD STARTS, WARM STARTS, AND CRASHES

Remember that the Aquarius computer can be restarted in two different ways. Pressing **RST**, then **RTN** initiates a "cold start"; you erase whatever program data is in the computer's memory.

Pressing **RST** followed by **CTL-C** causes a "warm start"; data in memory is not erased, unless the system has crashed. Remember that a crash is when Aquarius stops working, as though it has had a nervous breakdown. Electrical problems can cause a crash; or, Aquarius may simply not know how to respond to certain sequences of keyboard commands. A warm start or cold start will usually get Aquarius going again.

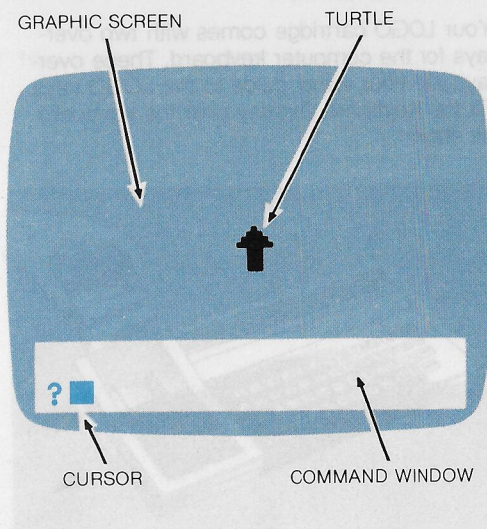
## ADD OVERLAYS

Your LOGO cartridge comes with two overlays for the computer keyboard. These overlays are your visual guide to the LOGO keys. Fit the Keyboard Overlay over the keyboard as shown.



## LET'S BEGIN

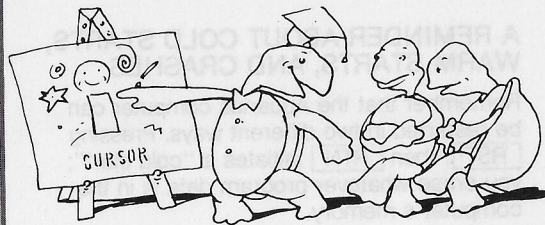
Be sure your computer is properly set-up, then insert the LOGO cartridge into the cartridge port. Now turn the computer ON and press the **RTN** key. The following screen should be displayed on your TV:



You see two areas on the TV screen. The larger area will display the graphic designs you create. This is called the Graphic Screen. The smaller area at the bottom of the screen will display exactly what you type on the Aquarius Keyboard. This area is called the Command Window.

The black square is the CURSOR, which appears in the Command Window. It marks the next position on the screen where a letter, number, symbol or space will appear when you press a key on the keyboard. The cursor helps you keep track of "where you are" on the screen. The "?" appears whenever the Aquarius is ready to receive your instructions.

**NOTE:** You will use the entire TV screen when you type information in Edit, List Processing, and Math. You can also do Math in Command Window.



## MODES

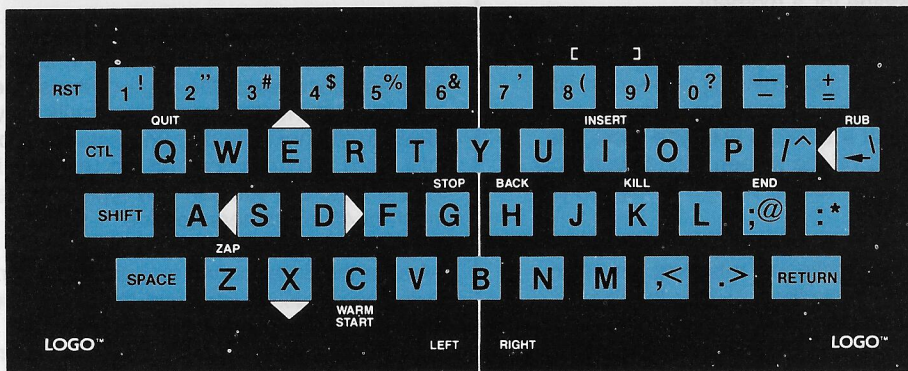
You can use your Aquarius LOGO in two different ways, or modes. In the IMMEDIATE Mode (or the LOGO Command Mode), Aquarius responds immediately to each instruction you type, as soon as you press the **RTN** key.

In PROGRAMMED Mode you will enter lists of instructions and the computer will not execute them until you tell it to do so. These lists of instructions are called "procedures"

## THE KEYBOARD

If you have never used a computer before, you will need to become accustomed to the keyboard.

On the computer keyboard you will see the letters of the alphabet, numbers, punctuation marks, and other symbols. Look familiar? The




computer keyboard is SIMILAR to that of a typewriter keyboard. However, there are some important DIFFERENCES. For example, be careful not to type the number 0 for the letter O, or the number 1 for the letter L. Though they may look alike, these keys send entirely different messages to the computer!

Another important difference between your computer keyboard and a typewriter is the special keys marked SPACE, CTL, RST, RTN, and the arrow keys...but more about that later.

Now type your name and press **RTN**. Type your address and press **RTN**. Ignore the Error Message for a moment.

Once you type in a line, press **RTN** and the cursor goes to the beginning of the next line. You will only see a maximum of four lines on the screen at a time. As you type more, the top line goes off the screen. You haven't lost lines that move off the screen — they're just out of sight!

**NOTE:** If you make any mistakes before you press **RTN**, use the  Key to move the cursor backward. Now correct the error. If you discover an error after you have pressed **RTN**, just retype the line correctly and press **RTN** once again.



### IMPORTANT:

Be sure you use the key you want!

Zero = 0

Letter O = O

One = 1

Letter L = L

### SPACE KEY/SPACING

Take another look at your computer keyboard. You'll see a key marked SPACE. When you press this key it will leave an empty space, just like regular typing.

Now try your hand at typing a complete sentence. Don't be concerned if the letters of your sentence reach the edge of the screen... keep typing! The cursor automatically moves down to the next line and understands if part of a word is on one line and the rest of it is on the next. But you can only type in a maximum of 45 characters per line.

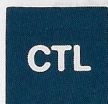
Most of the time you will include spaces in between words. As you read through these instructions, there will be times when you won't use spaces. It's important to type programs just as you see them — with spaces or without!

## SYMBOLS

Some keys on your computer keyboard will type two different things. For example, the number 1 key has an exclamation point (!) above it.

If you press a key which has two things on it, the lower symbol will appear on the screen. If you press the same key while holding down the SHIFT key, the upper symbol will appear on the screen.

## CONTROL — SPECIAL SYMBOL



The key marked CTL is called the CONTROL key. When used with another key, it allows you to display special symbols for Aquarius LOGO. To use the CTL key, always HOLD IT DOWN while pressing the other key — the same way you use the SHIFT key to type upper case characters.

When a key is to be used with the CTL key, the combination is written as CTL-(key). For example, CTL-8 displays an open bracket [ in the Command Window.

## RTN — THE ACTION KEY!



Look at the right side of the computer keyboard. You will see a key called RTN. (RTN is short for RETURN.) The RTN key is the key that makes things happen. After you type an instruction press the RTN key. In Immediate Mode, it tells Aquarius to execute the command you just gave. In Programmed Mode, it tells Aquarius to store the statement you just typed, so that it can be executed when you are finished writing your program. IF YOU DON'T PRESS THE RTN KEY AFTER TYPING IN EVERY COMMAND OR STATEMENT, NOTHING WILL HAPPEN.

The RTN key also sends the cursor to the beginning of the next line, so you're ready to type your next command or statement.

## RST KEY

**RST**

When you press the RST key, (short for RESET) you go back to the Aquarius title screen. If you press the RTN key after pressing RST, you erase whatever you previously input into the computer. The blank splitscreen is displayed. This is referred to as a "cold start".

**NOTE:** To learn about "warm starts", see page 7.

## HELPFUL HINTS

As you read the book, be on the lookout for the following helpful hints:

### ABBREVIATIONS/SPECIAL SYMBOLS

In many cases (noted throughout the text), the computer only responds to the Special Symbol or abbreviated forms of words for commands, and not the entire word. For ex-

ample, FD is the Special Symbol for FORWARD; BK means BACK; LT means LEFT TURN; RT means RIGHT TURN; CS means CLEARSCREEN. (See Appendix 0 for the full list of these Special Symbols.)

### ERROR MESSAGES

What happens if you make a mistake when typing a command and press the RTN key? The computer responds with one of any of the following messages, depending on the error made: "TOO FEW INPUTS...NEAR...", "SOMEONE DOESN'T LIKE...NEAR...", "I DON'T KNOW HOW TO...NEAR...", "I DON'T UNDERSTAND." The procedure name following NEAR identifies the procedure where the error is located.

Sometimes the Error Message is associated with a number, the exact meaning of these numbers listed in Appendix 0. No cause for worry... simply find your mistake in the line you just typed. Then retype the command correctly.

**NOTE:** Aquarius allows you to operate LOGO with no additional equipment. However, there may be times when the "NO ROOM" message appears. Refer to Chapter 7 and the Reference Section for suggestions on handling the NO ROOM message.

## CHAPTER 2

# GETTING STARTED WITH TURTLE GRAPHICS



### REMINDER...

before starting Chapter 2, be sure your computer is turned on and ready to go.

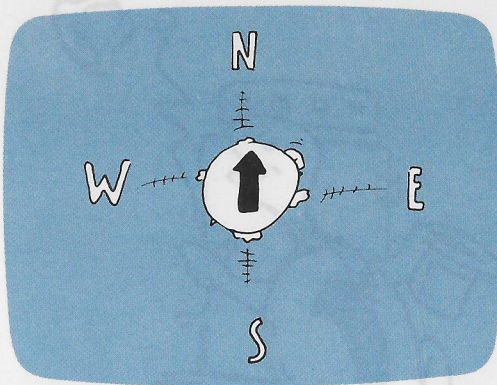
Welcome to the world of turtle graphics! Are you ready to create designs?

The Turtle (shown as an ARROW) is in the center of the screen, in the HOME position pointing upward (north). You will make drawings on the screen by sending messages to the Turtle. Give the Turtle commands and it leaves a trail as it moves. It's easy to create designs and drawings with Turtle graphics.



## TAKE THE TURTLE FOR A WALK

For a moment, think of your computer screen as a compass: the top is north, the bottom is south, the right is east, and the left is west. The Turtle is headed in whichever direction the ARROW is pointed, in this case upward (north).



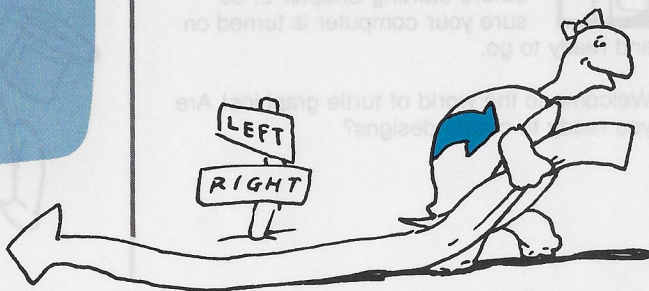
**NOTE:** This compass is specially designed for Aquarius LOGO!

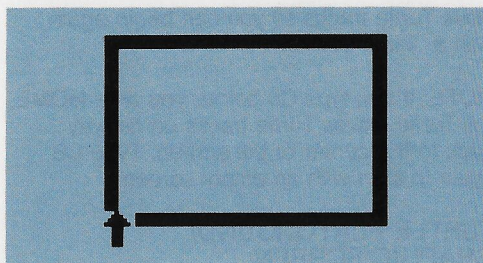
Send the Turtle a message to draw a box. You need to tell the Turtle which direction to go and how many Turtle steps to take. For example, `FD 10` tells the Turtle to go 10 steps in the direction the arrow is pointed. `BK 10` tells the turtle to go 10 steps in the opposite direction that the arrow is pointing. You can also tell the Turtle to turn **RIGHT** or **LEFT**. Each turn needs a number telling how many degrees to turn. For example, `RT 90` means make a right turn of 90 degrees.



### REMINDER...

remember that in some commands, the computer only responds to the Special Symbols for a command and not the entire word. For example, `FD` is FORWARD, `BK` is BACK, `RT` is RIGHT TURN, and `LT` is LEFT TURN.





Now type the following and watch your screen to see what happens:

FD 10 RT 90	<input type="text" value="RTN"/>
FD 10 RT 90	<input type="text" value="RTN"/>
FD 10 RT 90	<input type="text" value="RTN"/>
FD 10 RT 90	<input type="text" value="RTN"/>

These Turtle commands create the drawing as illustrated above. Remember to set the angle of the turn with RT or LT and set the number of steps with FD or BK. Each of these commands needs a number. Without it, the Turtle doesn't know how many steps to take or how many degrees to turn. If the number is missing, the Turtle doesn't move and you see the Error Message "TOO FEW INPUTS." Remember to leave a space between the command and the number.



### REMINDER...

press the  key after typing in each command, otherwise nothing will happen. Aquarius LOGO gives you a choice. You can type more than one command on a line.

### CORRECT YOUR TYPING

As you type LOGO commands, you might make typing mistakes.

Don't be concerned about making mistakes... part of the programming routine is learning to "debug" or fix a program.

OOPS! If you make any mistakes before completing a command, move the cursor backward for corrections with the  key. If you discover an error after you have typed the command and pressed , simply retype the line correctly and press  once again.

What happens if you leave out the necessary space and you type FD10? The Turtle sends you an Error Message "I DON'T KNOW HOW TO FD10?" This Error Message lets you know that the command you just typed needs special attention. (See Appendix 0 for listing of ERROR MESSAGES & their corrections.) The Turtle only understands messages that

are typed with the command first, followed by a SPACE, and then a number.

Now try typing FD10. Did the Turtle send you the same message? What needs to be changed? Did you add a SPACE and type in FD 10?

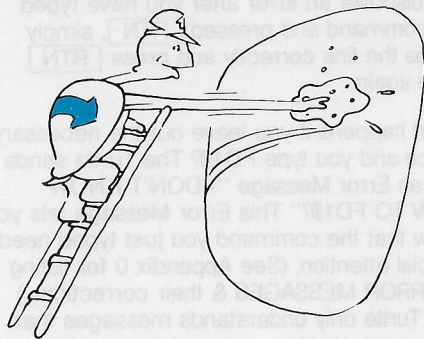
## CLEAR THE SCREEN

If you are ready to clear the screen and start over, type:

HOME [RTN]

CS (abbreviation for CLEARSCREEN) [RTN]

The HOME command sends the Turtle to its starting position (center of the screen) pointing up (north). The CS command erases pre-



vious Turtle tracks so you can begin again with a "clear screen".

**NOTE:** If you type CS before you type HOME, the Turtle leaves Turtle tracks on its way back to the center of the screen. Type CS again to start with an empty screen.

## TURTLE PLAYGROUND/ GRAPHIC SCREEN

Before going any further, it is important to learn the boundaries of your screen (the Turtle's playground/graphic screen). From the HOME position, the Turtle can take 30 steps North and 29 steps South. From the HOME position, the Turtle can take 39 steps East and 40 steps West.

If you tell the Turtle to move FD 50 from HOME, the following Error Message appears "OUT OF BOUNDS". This means that you have exceeded the Turtle's playground/graphic screen. Now, type:

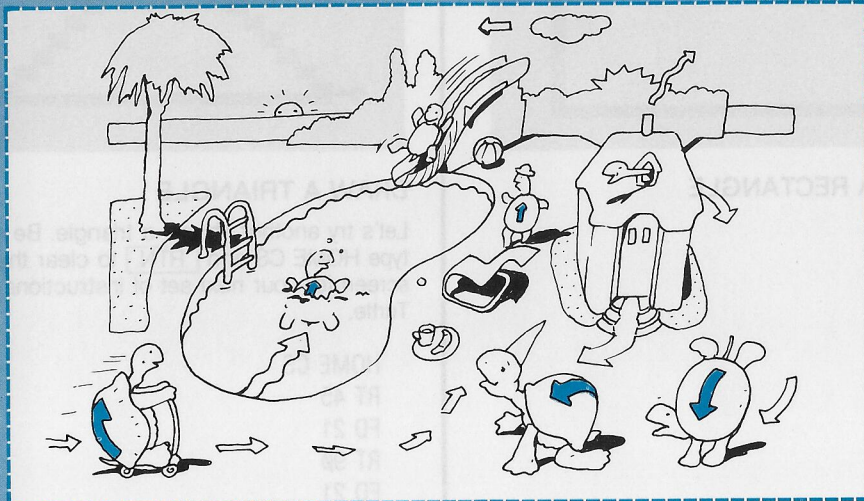
HOME CS

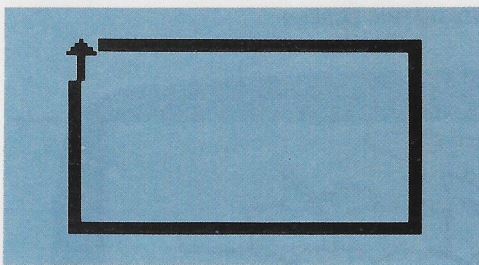
FD 30

[RTN]

Try other shapes: a rectangle, a box and a triangle.

BOUNDARIES OF TURTLE'S PLAYGROUND/GRAPHIC SCREEN

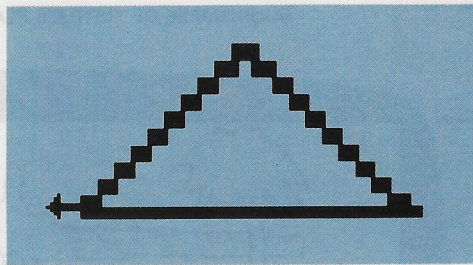




### DRAW A RECTANGLE

```
RT 90
FD 28
RT 90
FD 20
RT 90
FD 28
RT 90
FD 20
```

What makes a rectangle? A rectangle is a box with four sides...two long sides and two short sides. Our Turtle began its walk by making one of its four corners RT (Right Turn) 90 degrees. It drew two sides 28 steps long and two sides 20 steps long.

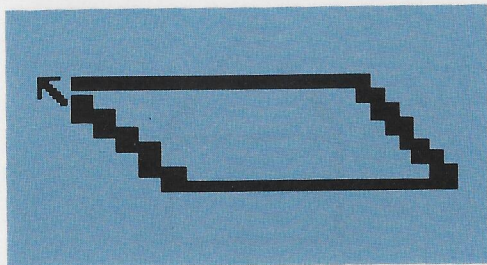


### DRAW A TRIANGLE

Let's try another shape, a triangle. Be sure to type HOME CS and RTN to clear the screen for your next set of instructions to the Turtle.

```
HOME CS
RT 45
FD 21
RT 90
FD 21
RT 135
FD 30
```

OOPS! What happened to your design? There's nothing wrong with your computer. The Turtle gets a case of the hiccups whenever it draws diagonal lines!



## DRAW A PARALLELOGRAM

How about drawing a parallelogram? A parallelogram is similar to a rectangle, except that each corner is not 90 degrees. Type:

```
HOME CS
RT 90
FD 20
RT 45
FD 10
RT 135
FD 20
RT 45
FD 10
```

Now try making some of your own designs with the Turtle. Be sure to keep a log of your drawings and designs, writing down the exact order of the commands so they can be drawn again quickly and easily.

## COLOR THE TURTLE/ BACKGROUND

In Turtle Graphics, you can draw with the Turtle just as you would with a pen or a pencil on paper...you decide both the color of the Turtle and the color to be used in your designs. The Turtle has 16 pen colors and 16 background colors to choose from:

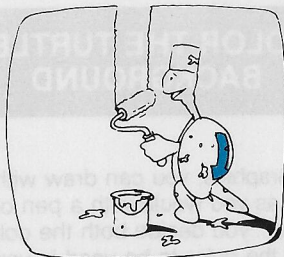
### COLOR CODE LIST

COLOR	CODE	COLOR	CODE
Black	0	Light Gray	8
Red	1	Blue-Green	9
Green	2	Magenta	10
Yellow	3	Dark Blue	11
Blue	4	Light Yellow	12
Violet	5	Light Green	13
Light Blue-Green	6	Orange	14
White	7	Dark Gray	15

To change the background color of your design, type the command PAPER, the number of the color you choose, and press the **RTN** key

PAPER 4

**RTN**



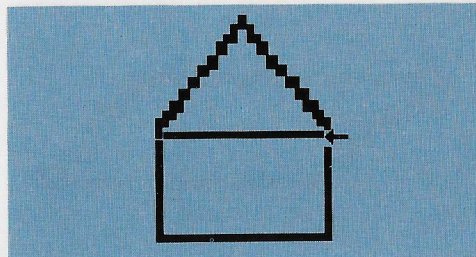
You can also change the color of the Turtle and its Turtle tracks. Type the command INK, the number of the color you choose, and press the **RTN** key.

INK 2 **RTN**

Okay! You're on your own...send the Turtle to its starting position (HOME), clear the screen (CS), and type in the Turtle commands to draw a new box (see page 15). Change the background color of your new box (PAPER). Now change the color of your Turtle and its tracks (INK).

## DRAW A HOUSE

OKAY! With this new trick of changing the Turtle's color, we can draw a house with a red roof and white walls. INK 7 changes the Turtle's color and Turtle Track to white. INK 1 changes the Turtle's color to red. All we need are the instructions for drawing a rectangle.



Type:

HOME CS

INK 7

RT 90

FD 28

RT 90

FD 20

RT 90

FD 28

RT 90

FD 20

INK 1

RT 30

FD 28

RT 120

FD 28

RT 120

Making a triangle for the roof is a bit tricky. The sides of the roof are 28 steps long, however if you measure the inside angles, each one is 60 degrees, or a total of 180 degrees.

## ERASE THE TURTLE'S TRACK

Sometimes you may want to change a line you've drawn. Instead of clearing the screen (CS command) and starting over, you can use the TILE 160 command in combination with FD or BK command to erase the Turtle Tracks. The idea is to change the Turtle TILE to a blank. Next you move the Turtle back to the spot where you wish to start drawing again. Type TILE 255 to draw your Turtle Tracks.

For example, draw a STAR, 26 Turtle steps long for each side. Type:

```
INK 0 PAPER 12
```

```
HOME CS
```

```
FD 26
```

```
RT 144
```

```
FD 26
```

```
RT 144
```

```
FD 26
```

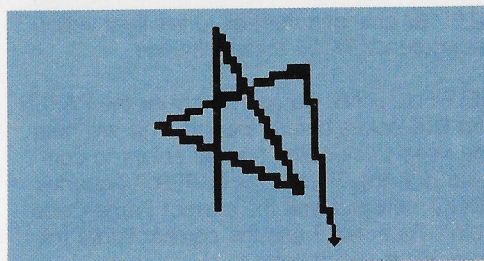
```
RT 144
```

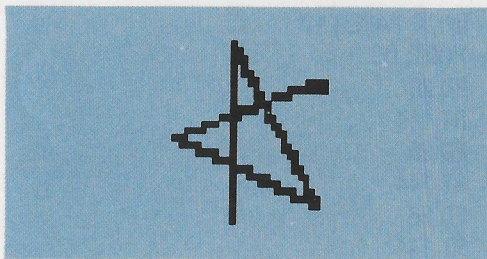
```
FD 26
```

```
RT 44
```

```
FD 26
```

OOPS! The last Right Turn (RT) is 100 degrees short, causing the STAR to look like this:





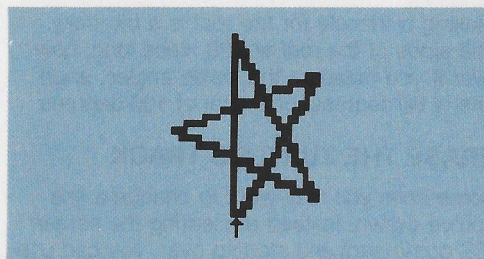
Let's erase the last Turtle Track, and return to the point where it made the wrong turn. To erase the last Turtle Track, type:

TILE 160  
BK 26



TILE 160 changed the trail to the light yellow background. BK 26 erased the line.

**HELPFUL HINT:** Be sure to check the PAPER and INK colors before erasing. This will help you keep track of the original colors to continue drawing. If you type PAPER ? Aquarius LOGO will give you the correct Paper Code Color. Type INK ? and the correct Turtle Ink Color Code will appear.



Now complete the STAR by typing:

TILE 255  
RT 100  
FD 26

TILE 255 changed the trail back to the original black line. RT 100 and FD 26 draws the line to complete the STAR.

## SAVE TYPING TIME USING REPEAT COMMAND

Now try drawing a square, using a shortcut method. To save time typing the same command or list of commands more than once, use the REP (short for REPEAT) command. The REP command takes three steps:

1. Type in the REP command.

**2.** Type in the number of times you want the Turtle to repeat the command.

**3.** Enclose the command(s) to be repeated in square brackets. (Make an open bracket by pressing the CONTROL key and the number 8 key; to close the brackets at the end of your command, press the CONTROL key and the number 9 key.) Make sure you typed brackets and not parentheses!

Ready? Type:

REP 4 [FD 15 RT 90]

RTN

Let's compare this command with the same BOX command we used earlier.

Original Command	Using REP Command
------------------	-------------------

FD 15  
RT 90  
FD 15  
RT 90  
FD 15  
RT 90  
FD 15  
RT 90

REP 4 [FD 15 RT 90]

Notice how the REP command saves time. Did you remember to put a space between the commands and the numbers (FD and 15, RT and 90)? If you didn't, Aquarius responds with the Error Message "I DON'T KNOW HOW TO". Simply retype the line correctly, leaving the appropriate spaces, enclosing the repeated commands in brackets, and press RTN once again.

How can we shorten some of the earlier programs you drew? Study the STAR program. Count the number of FD and RT commands we used. Ask the Turtle to follow the same set of instructions five times. This saves typing time. We can reduce the instructions to one line by using REP. Type:

REP 5 [FD 26 RT 144]

RTN

## SUMMARY

Chapter 2 introduced you to some of the basic commands in Turtle Graphics. Here's a short table to review these commands.

COMMANDS	DESIRED ACTION
FD number	FORWARD
BK number	BACK
LT number	LEFT TURN
RT number	RIGHT TURN
HOME	MOVE TURTLE TO CENTER OF SCREEN FACING NORTH
CS	CLEAR SCREEN WITHOUT MOVING TURTLE
PAPER number	BACKGROUND COLOR
INK number	COLOR OF TURTLE AND TURTLE TRACKS
REP number[action]	REPEAT A PROCEDURE A SPECIFIED NUMBER OF TIMES
TILE number	TRAIL LEFT BY THE TURTLE TRACKS



## CHAPTER 3

# MORE ABOUT TURTLE GRAPHICS

As you become more acquainted with Aquarius LOGO, you'll learn some exciting features which make it easier to move the Turtle around the Turtle Playground/Graphic Screen. In Chapter 3 you'll learn about the states of the Pen, states of the Turtle, simple direction commands, position commands, and a special Turtle Message command.

What if you want to move the Turtle to another place on the screen without leaving Turtle tracks that may ruin your design?

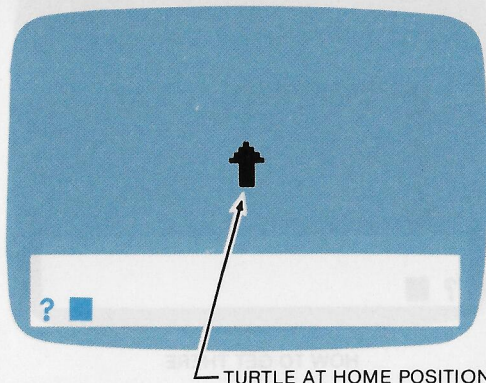
There is a command that you can use to move the Turtle without leaving a trail. Type:

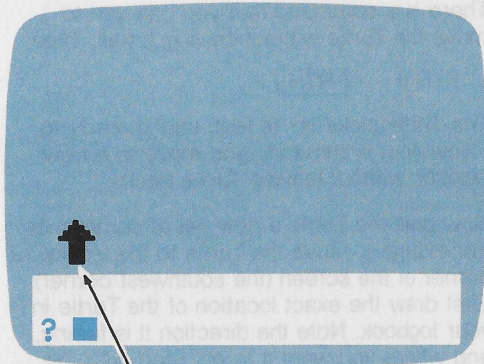
PEN Ø

RTN

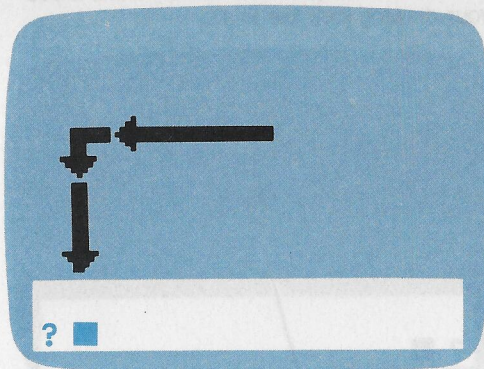
The Turtle picks up its feet, and is ready to follow your commands and move to a new position without leaving Turtle tracks.

Now give the Turtle a new set of commands. For example, move the Turtle to the left lower corner of the screen (the southwest corner). First draw the exact location of the Turtle in your logbook. Note the direction it is facing, and where you want it to go. Start with the Turtle at the HOME position. Be sure to type CS to erase the previous Turtle tracks. Does your drawing look like this?





MOVE TURTLE HERE



HOW TO GET THERE

```
LT 90 FD 40 LT 90 FD 25 RT 180
```

RTN

The Turtle walks to its new location, without leaving a trail, and is ready to draw. Type:

PEN 1

RTN

To begin drawing again (leaving Turtle tracks), type:

```
RT 90 FD 50
```

RTN

PEN 1 tells the Turtle to put the PEN down so the track is now visible.

## STATES OF THE TURTLE

Often in Turtle Graphics, you may want to make the Turtle invisible so it's not in the way of your designs. There is a command that allows you to "hide" the Turtle. Type:

Turtle 0

RTN

The Turtle becomes invisible. No other commands are affected. You can still move the Turtle around with FD, RT, LT. You won't see the Turtle, but you can see its trail. To make the Turtle visible, type:

Turtle 1

RTN

## TURTLE SHORTCUTS

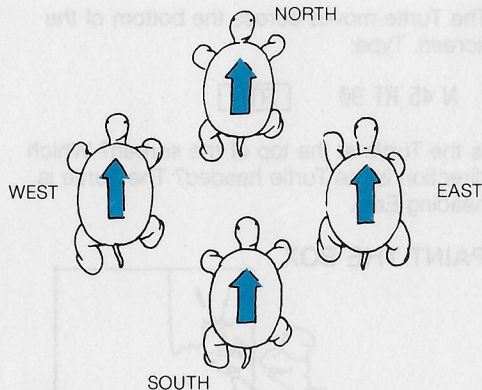
Here are a few shortcuts. For example, there are two ways to move the Turtle to a new position: using "directional" commands and a "positional" command.

### DIRECTIONAL COMMANDS

Directional commands allow you to move the Turtle:

E for east      N for north  
W for west      S for south

Remember your compass? HOME is your starting point with the Turtle facing north.



Use these directional commands to move the Turtle to another place on the playground/ graphic screen without having to change its heading.

E 35      RTN

The Turtle moves to the right side of the screen 35 Turtle steps. The Turtle faces in the same heading as before. Type:

S 20      RTN

The Turtle moves down the screen. Notice the direction the Turtle is facing. Type:

W 60      RTN

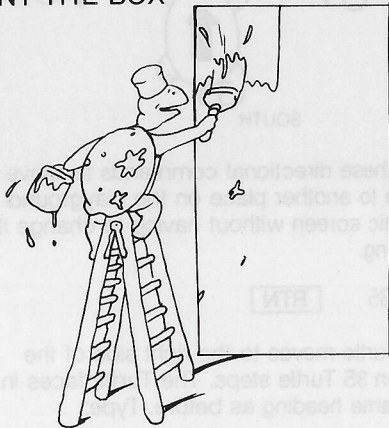
The Turtle moves across the bottom of the screen. Type:

N 45 RT 90

RTN

Is the Turtle at the top of the screen? Which direction is the Turtle headed? The Turtle is heading East.

### PAINT THE BOX



Directional commands are especially useful when drawing color blocks. By stacking a number of lines, one on top of the other, the Turtle leaves behind a painted BOX or color block. Type:

HOME CS

INK 1

E 20

N 1

W 20

N 1

E 20

Can you see what is happening? First the Turtle moved E (east) 20 steps, moved up a step (N 1), and then back 20 steps (west), placing a line on top of the first line.

Let's take advantage of the REP (repeat) command to draw a color block. Remember to include an instruction to go in one direction, up one step, and back in the opposite direction the same number of Turtle steps. One instruction will give you a solid stacked line. The repeat command will give you the width of the block. In our example, the block is 30 Turtle steps long and 20 Turtle steps wide.

REP 20 [E 30 N 1 W 30]

RTN

How about painting a parallelogram? The secret is to follow the same process with a slight change. When stacking the lines one on top of the other, extend the block East or West a short distance. Type:

```
HOME CS [RTN]
E 25
S 1
E 1
W 25
E 25
S 1
W 1
W 25
```

A line, 25 Turtle steps long, begins the stack. The line is extended East one (1) Turtle step each time the stacking takes place. To paint a parallelogram 25 Turtle steps long and 10 Turtle steps high, type:

```
HOME CS [RTN]
REP 10 [E 25 S 1 E 1 W 25] [RTN]
```

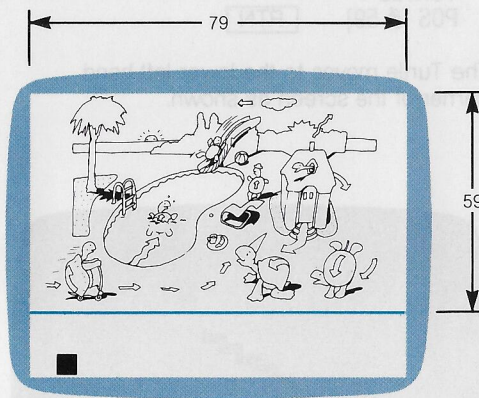
Type Turtle  $\emptyset$  [RTN] to hide the Turtle and see the total design.

Now when you draw your pictures using the directional commands, you can begin your designs anywhere on the screen.



### REMINDER...

these are the limits of your playground/graphic screen.



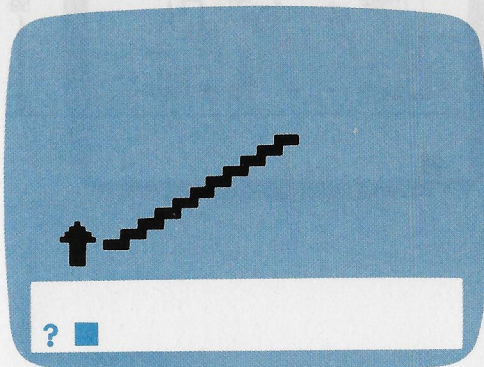
## POSITIONAL COMMAND

The positional command POS is a convenient way to move the Turtle immediately to a new spot on the playground/graphic screen. Type:

TURTLE 1  
POS [0 59]

RTN

The Turtle moves to the lower left-hand corner of the screen as shown.



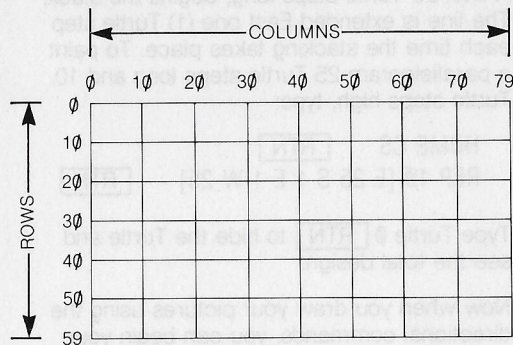
Did the Turtle leave a trail behind it? Type CS to erase the trail.

The numbers inside the square brackets stand for column and row numbers. Every point on the graphic screen is represented by a column and row number. Look at the graph Playground/Graphic Screen below. The middle of the graphic screen, or Home position, is column 40 and line 30. There are a total of 80 columns (0 to 79) and 60 rows (0 to 59) on the graphic screen. Can you locate column 79 and row 59? If you said the bottom righthand corner, you're right. Try it. Type the following command, and be sure to enclose the numbers with brackets.

POS [79 59]

RTN

The Turtle moves to that location immediately.



Where is the Turtle headed?

To identify the Turtle's current location type:

**POS ?**

The Turtle responds with the column and line number of its location.

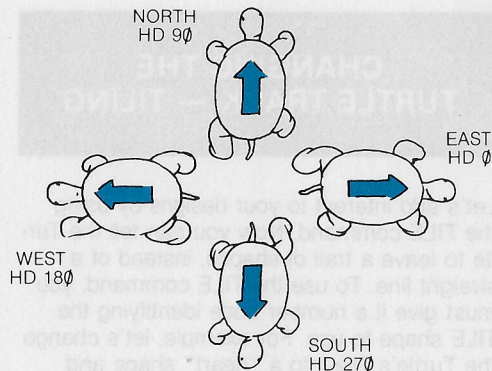
Now practice placing the Turtle anywhere on the screen. Keep track of your instructions by writing them down in your logbook.

## HD COMMAND

There will be times when you won't know how many degrees the Turtle has made in a turn. To help you get your "bearings", use HD, the heading command, to tell you how many degrees the Turtle is currently facing from 0 (zero). Here...zero degrees is when the Turtle is facing east. Type:

**HD ? (question mark)**

The Turtle responds with the number of degrees it is facing.



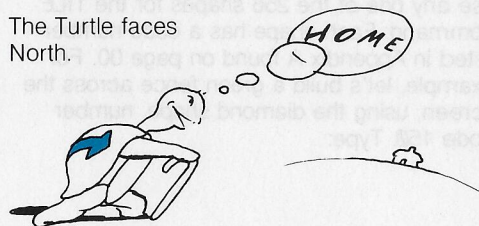
To head the Turtle in a new direction, type:

**HD 0 (zero)**

The Turtle now faces the new direction, East.  
Type:

**HD 90**

The Turtle faces North.



## CHANGING THE TURTLE TRACK — TILING

Let's add interest to your designs by using the TILE command. Now you can tell the Turtle to leave a trail of shapes, instead of a straight line. To use the TILE command, you must give it a number code identifying the TILE shape to use. For example, let's change the Turtle's Tile into a "heart" shape and have it draw a BOX. Type:

```
HOME CS      [RTN]  
TILE 212  
REP 4 [FD 20 RT 90]
```

Notice the BOX is formed by a string of "hearts" rather than a straight line. You can use any one of the 256 shapes for the TILE command. Each shape has a code number listed in Appendix A found on page 00. For example, let's build a green fence across the screen, using the diamond shape, number code 150. Type:

```
HOME CS      [RTN]  
INK 2  
TILE 150  
PEN 0  
POS [10 40]  
PEN 1  
REP 15 [E 60 N 1 W 60]
```

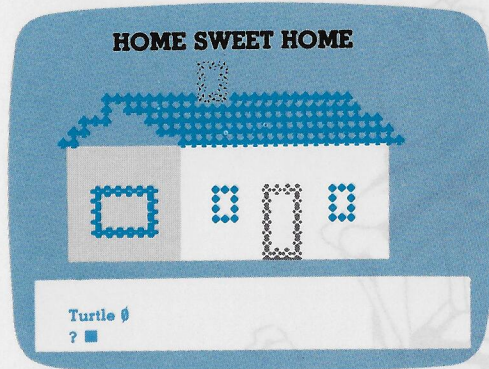
To return to the original Turtle Tracks, type TILE 255 and press [RTN].

## NAME YOUR DRAWING WITH TURTLE PRINTS

Sometimes the drawings you make in Turtle Graphics are so creative that you will want to give them a name. You can do this by typing in the TP (short for TURTLE PRINT) command, which takes two steps:

1. Move the Turtle to the place where you want the 'message' printed. (Be sure to use PEN 0 for pen-up if you don't want to leave Turtle tracks along the way.)
2. Once you select your position, type:

TP [space] ' (an apostrophe) [space] and then your message. The Turtle leaves your message right where you place it.



## DISCUSSION ON HOME SWEET HOME

Let's put a caption HOME SWEET HOME on the screen and use what we learned to draw a house with a red roof, white walls, blue windows, and a green door. Ready?

Begin by placing the caption in red, near the top of the Turtle's Playground/Graphic Screen.

```
HOME CS      [RTN]
PEN 0
POS [25 2]
INK 1
TP ' HOME SWEET HOME'
```

Next we move the Turtle to a new location and draw the front wall of the house in solid white.

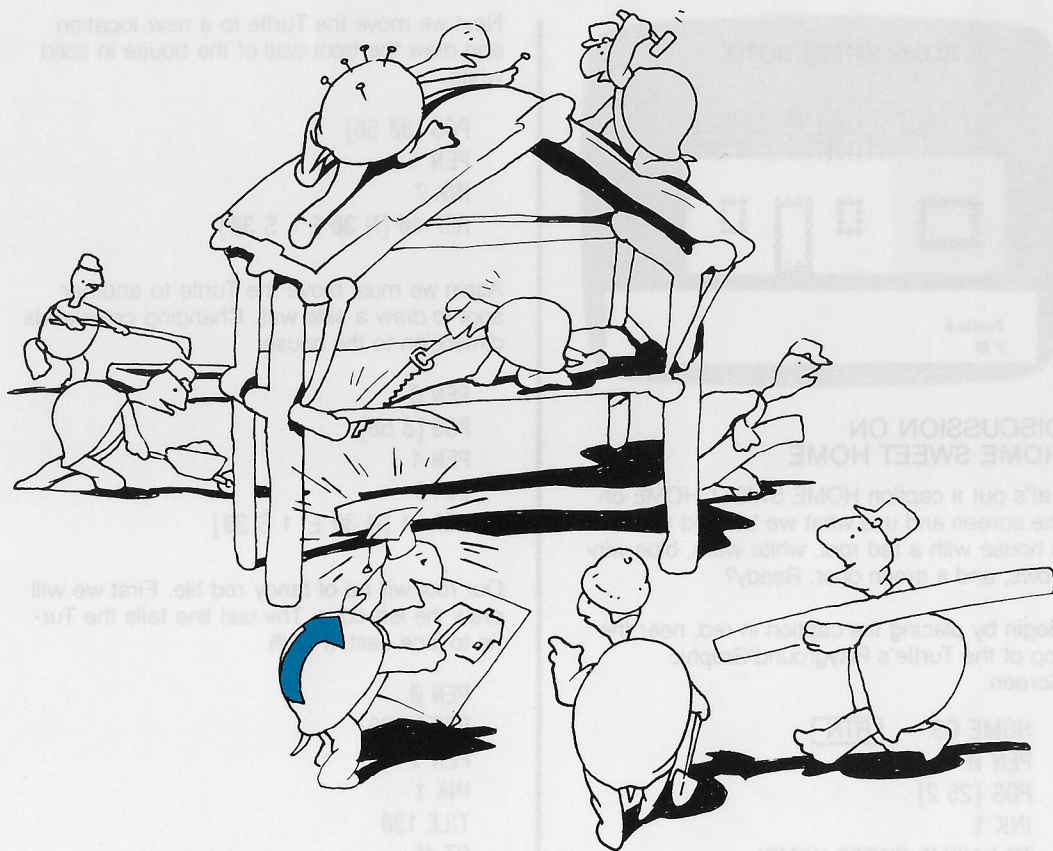
```
POS [30 56]
PEN 1
INK 7
REP 40 [N 30 E 1 S 30]
```

Again we must move the Turtle to another spot to draw a side wall. Changing color adds dimension to the house.

```
PEN 0
POS [8 56]
PEN 1
INK 8
REP 21 [N 30 E 1 S 30]
```

Our roof will be of fancy red tile. First we will draw the left edge. The last line tells the Turtle to face east, (HD 0).

```
PEN 0
POS [8 26]
PEN 1
INK 1
TILE 138
RT 45
```



FD 17  
HD Ø



To finish our roof, we will draw a solid parallelogram.

REP 6 [E 4Ø S 2 E 2 W 4Ø]

Doesn't the house look great? Let's complete it by adding windows, a door, and a chimney. Each set of instructions has two parts: one set of commands to move the Turtle to a new spot, and the other to begin drawing again.

For one window, type:

PEN Ø  
POS [15 38]  
PEN 1  
INK 4  
TILE 23Ø  
REP 4 [FD 1Ø RT 9Ø]



To make the door, type:

PEN Ø  
POS [46 38]  
PEN 1  
INK 2  
TILE 213  
REP 2 [FD 6 RT 9Ø FD 18 RT 9Ø]

Let's make two small windows on each side of the door. First, the left one. Type:

PEN Ø  
POS [36 38]  
PEN 1  
INK 4  
TILE 135  
REP 2 [FD 4 RT 9Ø FD 8 RT 9Ø]

Next, the right one:

PEN Ø  
POS [58 38]  
PEN 1  
REP 2 [FD 4 RT 9Ø FD 8 RT 9Ø]

To add the chimney, type:

```
PEN Ø  
POS [35 7]  
PEN 1  
INK Ø  
TILE 2Ø9  
REP 2 [FD 4 RT 9Ø FD 6 RT 9Ø]
```

Move the Turtle out of the way to complete the Picture.

### TURTLE Ø

Now that you know how to draw a house, try some of your own designs. Always include a 'move' routine to help the Turtle move from one spot to the next. Be sure to keep a log of all your design notes and drawings. Don't expect your projects to run perfectly the first time. The fun in LOGO is solving the problem.



### REMINDER...

be sure to type PEN 1 for pen-down to start drawing again.

When you finish using Turtle Graphics, turn your computer off and the Turtle along with all your creations will disappear.

Finished using the computer for now? Then turn off the computer. Remove the Aquarius LOGO cartridge.

But if you are ready to continue with bigger and better things, leave the cartridge in, the computer turned on, and begin the next chapter!



## SUMMARY OF TURTLE COMMANDS & ABBREVIATIONS

Chapter 3 introduced you to many of the additional concepts and procedures found in using the Turtle Graphics mode of LOGO. For your review, here are the commands you learned in this section:

COMMANDS	DESIRED ACTION
HD number	HEADS THE TURTLE IN A NEW DIRECTION
PEN $\emptyset$ (up)	NO TURTLE TRACKS MADE
PEN 1 (down)	TURTLE TRACKS MADE
DIRECTION COMMANDS	DESIRED ACTION
E number = EAST	TURTLE MOVES SPECIFIED STEPS WITHOUT CHANGING DIRECTION IT IS HEADED
W number = WEST	
N number = NORTH	
S number = SOUTH	
POS [column row]	MOVE TURTLE TO ANOTHER PLACE ON SCREEN WITHOUT CHANGING THE DIRECTION IT IS HEADED
POS ?	IDENTIFIES TURTLE'S CURRENT LOCATION

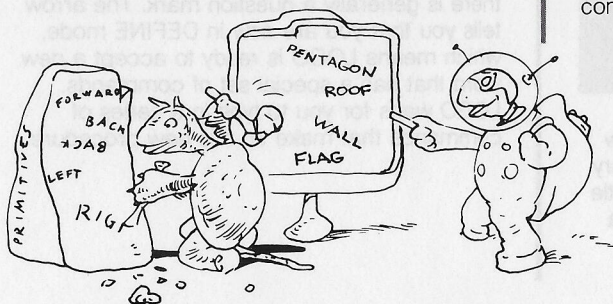
DIRECTION COMMANDS	DESIRED ACTION
TILE number	CHOICE OF TURTLE TRACKS TO CHANGE ITS TRAIL
TURTLE 0 (zero)	HIDES TURTLE: STILL LEAVES TRACKS BEHIND
TURTLE 1	SHOWS TURTLE: REAPPEARS
HD ?	NUMBER OF DEGREES TURTLE CURRENTLY FACING
TP ' message '	TURTLE PRINTS MESSAGE TO ACCOMPANY DRAWING

## CHAPTER 4

# CREATING A TURTLE VOCABULARY

By now you are probably discovering that the Turtle Graphics mode of LOGO can be fun and challenging. The LOGO language is similar to your daily conversation; for example, the vocabulary words you use to teach the Turtle to walk (FD, BK, LT, RT, HOME, and CS) are words “built-in” to the LOGO language. Those who developed LOGO call these words “primitives” because they are the words or commands the computer already knows, or has been taught. All the commands you have used so far have been primitives.

You can increase the Turtle's vocabulary by teaching the computer new words. Once you create a new word, it becomes part of the Turtle's vocabulary. Whenever it is used, the Turtle draws a specific design that you have created. A “procedure” is a command or series of commands that you teach the computer.



## CREATING A NEW TURTLE WORD USING PROCEDURES

Type the word:

PENTAGON

RTN

What happens? Does the Turtle move or did you receive the Error Message "I DON'T KNOW HOW TO PENTAGON." If you did, the Turtle is letting you know that up to now, PENTAGON is not a part of its existing vocabulary. It does not know how to do that command.

**? PENTAGON**

**I DON'T KNOW HOW TO PENTAGON  
? ■**

However, you can teach the Turtle this new word (procedure) and add it to its vocabulary list. As a result, whenever you give the Turtle the command PENTAGON, it will perform a sequence of commands and draw a five-sided figure.

To teach the Turtle a new word (procedure), start with a "name" (one that reminds you of what the procedure does). In this example, the word PENTAGON best describes the shape you want the Turtle to draw. To let the computer know that you are ready to teach it a new word, type TO before the name you've chosen for the procedure.

Type:

TO PENTAGON

RTN

**? TO PENTAGON  
→ ■**

Something happens in the text window. An arrow ( → ) appears on the screen where there is generally a question mark. The arrow tells you that you are now in DEFINE mode, which means LOGO is ready to accept a new word that has a special set of commands. LOGO waits for you to type in a series of commands that make up this new procedure.

Now type in the series of commands that make up the definition of your new word PENTAGON. Be sure your commands are spelled correctly and remember to include the number of Turtle steps or degrees, spaces between the commands and numbers, and to press **RTN** when you finish each line.

LT 90 REP 5 [FD 15 RT 72] **RTN**

Once you have typed in your special set of commands, you are ready to leave the DEFINE mode and return it to the Turtle's COMMAND mode. To leave the DEFINE mode, type the command END on the next line with the arrow. END tells the computer that the procedure is complete.

END **RTN**

LOGO responds with:

PENTAGON  
?

This tells you that the computer has placed the new word (procedure) into memory and added it to the Turtle's existing vocabulary. Now the Turtle will draw a PENTAGON whenever you give it this command.



### REMINDER...

you must always type END as the last command in a procedure.

Review the three steps to create a new word (procedure). Type:

### TO PENTAGON

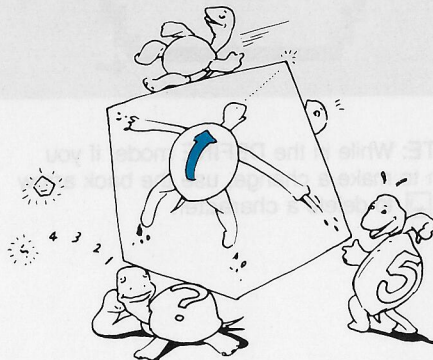
*Enter the DEFINE mode.*

LT 90 REP 5 [FD 15 RT 72]

*Define special set of commands.*

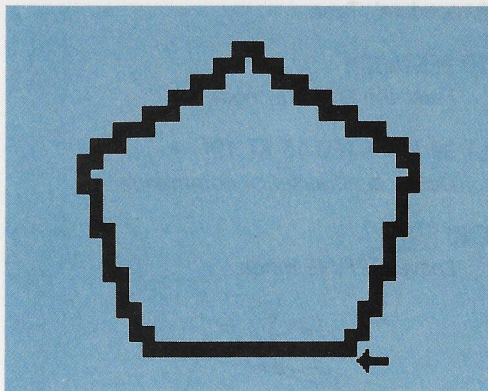
END **RTN**


*Leave DEFINE mode.*

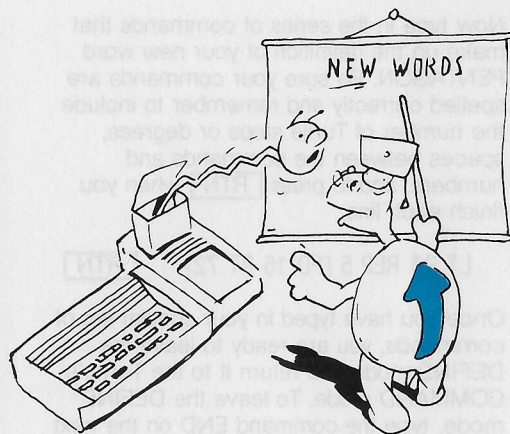


Ready to try your new word? Type:

HOME [RTN]  
CS [RTN]  
PENTAGON [RTN]



**NOTE:** While in the DEFINE mode, if you wish to make a change, use the back arrow key  to delete a character.



## BOX, RECTANGLE, STAR AS PROCEDURES

You are ready to teach your computer some new words. You will tell it how to draw the shapes we've talked about earlier, a BOX, RECTANGLE, and STAR.

### TO RECTANGLE

```
REP 2 [RT 90 FD 28 RT 90 FD 20]  
END
```

To teach the computer to draw a triangle 30 Turtle steps for each side with 60 degree angles, type:

```
TO TRIANGLE
LT 90
REP 3 [RT 120 FD 30]
END
```

Try writing your own procedure for a STAR.

## PASSING PROCEDURES

The procedure you've created or defined can be used as part of other procedures. Including the name of one procedure within another procedure is called Passing Procedures. A principle in LOGO programming is to break your final design into many small subprocedures, and then use a 'linker' program to tie them all together. This gives you the ability to use the same procedure in other programs. For example, the same RECTANGLE procedure can be used to draw a spinning rectangle called SPIN, and in another procedure, a simple house called SIMPHOUSE.

```
TO SPIN
HOME CS
REP 4 [RECTANGLE LT 90]
END
```

Type SPIN RTN

Spin is formed by drawing four (4) RECTANGLES. The Turtle then makes a left turn (LT) 90 degrees after each RECTANGLE is drawn.

```
TO SIMPHOUSE
HOME CS
INK 7
RECTANGLE
INK 1
TRIANGLE
END
```

Type SIMPHOUSE RTN

The simple house is drawn by placing the RECTANGLE and TRIANGLE procedure in one procedure.

## ERRORS IN TYPING PROCEDURES

If the Turtle sends you the Error Message "I DON'T KNOW HOW TO...NEAR..." you may have misspelled a word, left out a space between a command and number, typed the number 0 for the letter O or the number 1 for lower case L (l).

To make a change in your procedure, enter the EDIT mode. Type:

EDIT PENTAGON

RTN

This places you in the EDIT mode and the screen displays a listing of your procedure.



### REMINDER:

To make sure you are in the Edit Mode, see if the whole screen has turned blue and the line at the bottom reads — ESDX, l)nsert, Z)ap, K)ill, Q)uit. To enter Edit Mode if you are not there, type EDIT before the name of the procedure.

### TO PENTAGON

```
■ LT 90  
REP 5 [  
  FD 15  
  RT 72 ]  
END
```

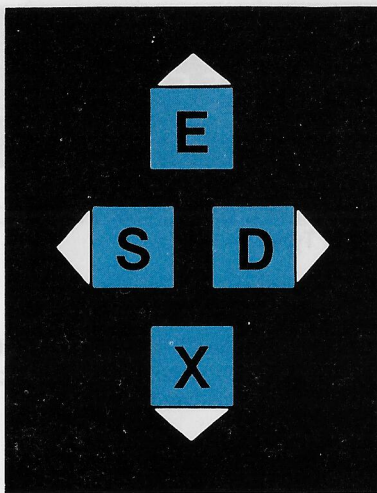
ESDX, l)nsert, Z)ap, K)ill, Q)uit

Look for the error you have made. Move the cursor to the line and/or letter(s) you will change. You can also take out (delete) any part of a word or add (insert) another letter or word.

### DELETE

To take out or delete a word or letter:

1. Move the cursor to the place where you want to delete, using the E, X, D, or S keys. To move the cursor UP to the previous line, press the **E** key; to move the cursor DOWN a line, press **X**; to move the cursor to the RIGHT, press **D**; to move the cursor to the LEFT, press **S**.



2. Press the **[Z]** key to delete the character under the cursor. Z stands for the word “zap”.

3. Press the **[K]** key to delete the current line. K stands for the word “kill”.

## INSERT

Inserting a character, word, or line of words takes two steps:

1. Move the cursor to the place where you want to make the addition, using the E, X, D, or S keys.

2. Press the **[I]** key to enter the INSERT mode. Type in your new letter or word. The characters behind the cursor are automatically pushed forward. Use the **[←]** (back arrow key) to make corrections. To leave the INSERT mode, press the **[;]** (semi-colon) key.

## CHANGE

Changing a character, word, or line of words takes three steps:

1. Move the cursor to the place where you want to make the change, using the E, X, D, or S keys.

2. Zap or delete the unwanted character, then press the **[I]** key to enter the INSERT mode. Type in your new letter or word.

3. Press the **[,]** key to leave the INSERT mode and return to the EDIT mode.

**NOTE:** If you insert a new line of instructions, be sure to leave a space between commands and press the **[RTN]** key upon completion. This minimizes the risk of confusing Aquarius.

Once you have completed all the changes you wish to make to your procedure, you are ready to leave the EDIT mode, by pressing the **[Q]** key which stands for Quit.

**NOTE:** For a more detailed explanation of the EDIT mode, see page 00 of the Reference Section.



## PRACTICE EXERCISES

Try modifying or changing your PENTAGON procedure. Type:

### EDIT PENTAGON

Your goal is to delete LT 90. Now press the **Z** key seven times to delete the character under the cursor. The line then reads:

REP 5 [FD 15 RT 72]

Press the **Q** key to place the change into the computer's memory.

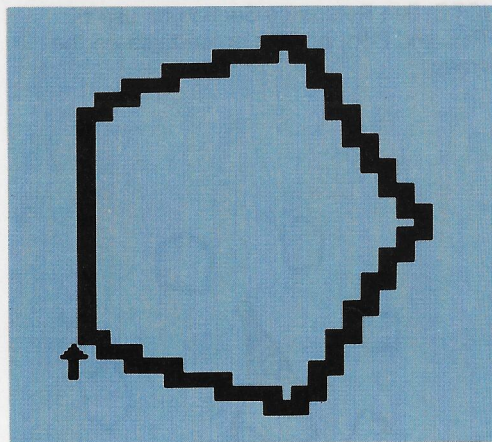
**ESDX, I)nsert, Z)ap, K)ill, Q)uit**

LOGO responds with:

?

Type:

PENTAGON



HERE'S ANOTHER PRACTICE EDIT.

Try flipping the pentagon over on its head.

Type:

EDIT PENTAGON

Your goal is to add or insert RT 90 before the REP command. Type the letter **I** to enter the INSERT mode. Type your addition, in this case:

RT 90

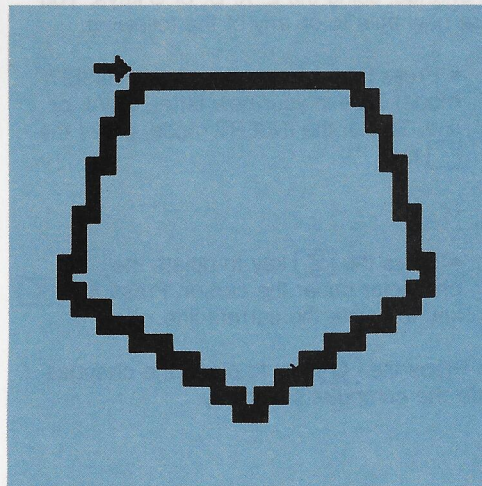
RTN

Press the **;** (semi-colon) key to exit the INSERT mode, followed by the letter **Q** to place your changes into the computer's memory. LOGO responds with:

?

Type:

PENTAGON



## SUMMARY

Review the EDIT commands you've learned in Chapter 4.

1. To enter the EDIT mode, type EDIT and the name of the command.
2. Move the cursor where you want to make a change, using the E, X, D, or S keys. You are now able to do any of the following:

- Press the **I** key to enter the INSERT mode. Type in your new letter, word, or line. To exit the INSERT mode, press the **;** key.

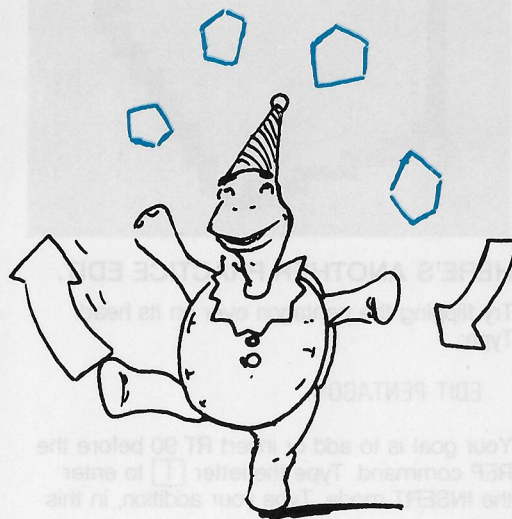
or

- Press the **Z** key to delete the character under the cursor. Press the **K** key to delete the current line.

3. Press the **Q** key to place the changes into the computer's memory.

4. Wait for the question mark (?) to appear before you continue in the Turtle Graphics mode.

**NOTE:** See Reference Section on use of FREE and P for printing procedures on the screen.



## CHAPTER 5

# EXPANDING YOUR TURTLE VOCABULARY USING VARIABLES

Before you begin this chapter, be sure to reset the computer or plug in a 4K or 16K cartridge. This is necessary because the following procedures use a lot of memory space! If you run out of space you will see the "NO ROOM" message.

Creating a LOGO vocabulary is easy using the DEFINE mode. You are now going to learn about increasing the power of LOGO in programming, using variables. A "variable" stands for a changeable value in a command. For example, instead of typing FD 10, the

number can be changed to FD :SIZE. :SIZE now stands for any number you choose in the FD command. :SIZE is called a variable.

There may also be times when you want to vary the size of your drawing. Rather than rewriting the procedure to change the size of the design, you can use the variable called :SIZE.

For example, the procedure for drawing a box is:

```
TO BOX
  REP 4 [FD 25 RT 90]
END
```

To add flexibility to your procedure, add the variable

```
TO BOX :SIZE
  REP 4 [FD :SIZE RT 90]
END
```

In LOGO, when you want a word to be a variable, you type a colon (called Dots in LOGO) followed by the name of the variable. In this example, :SIZE is the variable and is found in two places in the procedure.

1. On the first line, so that LOGO knows a variable follows.

2. Within the procedure, inside the brackets where it replaces a number after the FD command. This means you can enter any number for FD.

Now you have a flexible BOX procedure.

To run your program, type:

BOX 10

*(Designs a BOX with 10 Turtle steps for each side.)*

BOX 25

*(Designs a BOX with 25 Turtle steps for each side.)*

If you type BOX, LOGO responds with the Error Message "TOO FEW INPUTS". This means that the Turtle is looking for input from you (a number) telling how far you want it to move. Type:

BOX 30 [RTN]

Now see what happens!



### REMINDER...

be sure to type HOME and CS after you make your boxes and press [RTN] after each command.

Practice this procedure using different numbers or values. Copy the program into your logbook to help you remember how to write a procedure for BOX using variables.

## FUN WITH VARIABLES

Ready to teach the Turtle to draw boxes while changing their sizes automatically? Call this new word or procedure BOXES. It takes two sets of procedures. Type:

```
TO BOX :SIZE  
  REP 4 [FD :SIZE RT 90]  
END
```

Next type:

```
TO BOXES  
  BOX 10  
  BOX 20  
  BOX 30  
END
```

Wow! Okay — change the sizes. EDIT "BOXES and change the size to 5, 10, 15, 20, and 25.

Be sure to copy your program into your log-book and draw the picture for future reference.

As we've seen from the examples, a variable can represent numbers. But it can do more than that. Actually, you can use variables to represent any value found in other commands. For example, you can assign the Turtle's position, the column and row number, from another variable to the POS command. In the FLAG program, we will create a variable :PLACE to accept the different values for the POS command. It adds flexibility to your program, especially when you want to draw the same design in different locations. To see this in operation, type:

```
TO FLAG :PLACE
PEN Ø
POS :PLACE
PEN 1
FD 3Ø
REP 4 [RT 9Ø FD 12]
END
```

The Turtle's position has been assigned to :PLACE. This means each time you execute this program, the FLAG will be drawn in a different location. The procedure FLAG begins

with a pole, FD 3Ø, then a flag is drawn using the REP instructions. Type:

```
HOME CS [RTN]
FLAG [15 5Ø] FLAG [35 5Ø] FLAG [55 5Ø]
```

Notice that the numbers assigned to the variable :PLACE are enclosed in brackets. This assigns position numbers, the columns and rows, in the format the POS command can recognize.



#### REMINDER:

Do not be concerned if you reach the end of the Command

Window when typing this line. Just continue typing. The cursor will automatically advance to the beginning of the next line.

## REVIEWING VARIABLES

Here are some important points to remember about variables:

1. Variables are used when you want to give flexibility to a procedure by changing the size of your box or shape.

2. Dots (:) are used before each variable, with no space between the dots and the name of the variable (for example, :SIZE).

3. A variable requires a number or value (input) to the procedure. For example, BOX 10, or FLAG [35 50]. The value that is input must be consistent with the format that the command can recognize.

4. You can create any name you wish for the variable, as long as it is used within the procedure. For example, you can use any of the following words to represent the size of the side:

TO BOX :SIDE

TO BOX :LENGTH

TO BOX :WIDTH

TO BOX :X

The procedure to draw each side of the BOX could read:

FD :SIDE FD :LENGTH FD :WIDTH and/or  
FD :X

Now you have two choices in determining how you write your procedure:

TO RECTANGLE

REP 2 [FD 20 RT 90 FD 30 RT 90]

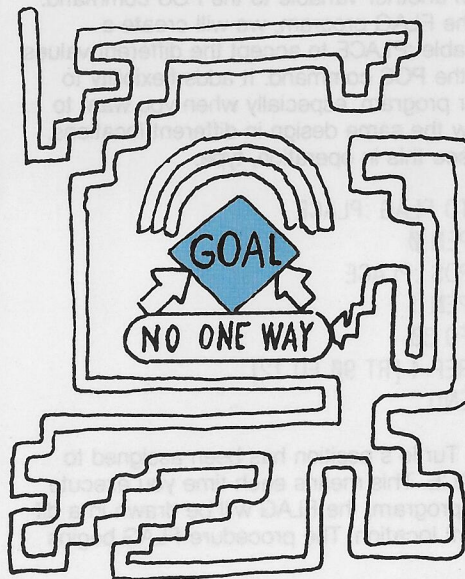
END

TO RECTANGLE :WIDTH :LENGTH

REP 2 [FD :WIDTH RT 90 FD :LENGTH  
RT 90]

END

One of the most enjoyable features about LOGO is that there isn't one correct way to write a procedure. There are many paths you can take to reach your goal.



## USING VARIABLES IN A PROCEDURE

Up until now you told the computer to draw a shape to a certain size. Now you can tell it to draw a shape and use a variable to represent the size. The variable stands for any size you want. The computer waits for you to input the size. Input means to type in the size. Each time the computer draws the shape, you can choose a different size!

You can create some interesting designs using variables in your procedure. LOGO procedures can be written to accept more than one input. Simply choose a name for each input, preceded by a colon. Type:

```
TO COLORFLAG :PLACE :COLOR
FLAG :PLACE
INK :COLOR
END
```

In COLORFLAG you can change the ink color each time you run the FLAG procedure by adding the INK color code number as a variable. To run the program, type:

```
HOME CS
COLORFLAG [ 15 50 ] 1
COLORFLAG [ 55 50 ] 4
```

A red and blue flag will appear on your Graphics Screen. Be sure to use brackets [ ] and spaces between each of the numbers for the variable :PLACE.

## MAKING AN ARC OR A CIRCLE

In LOGO Turtle Graphics, a circle is drawn by moving the Turtle forward a small distance, and turning a small angle each time over and over until a circle 360 degrees is formed.

Type:

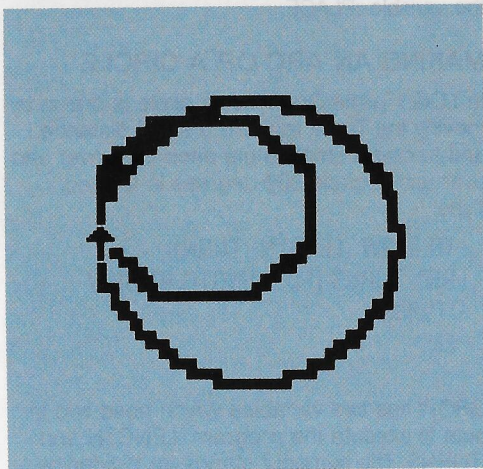
```
TO GROW :LENGTH :TURNS
REP :TURNS [FD :LENGTH RT 360/
:TURNS]
END
```

GROW has two variables which need two inputs to execute the program :LENGTH and :TURNS. The symbol / stands for divided-by. Notice that the bigger the number we assign to :TURNS, the smaller the angle the Turtle will turn. Be sure to **RST** if you are not using a 4K or 16K memory cartridge.

Now type:

```
HOME CS
GROW 1 120
GROW 10 8
```

Your screen will look like this:



GROW 1 120 draws a circle while GROW 10 8 draws an Octagon. The difference between the two programs is the number of Turtle steps taken and the angles turned.

#### COMMAND

#### WHAT HAPPENED

```
GROW 1 120
```

```
REP 120 [ FD 1 RT 3 ]
```

```
GROW 10 8
```

```
REP 8 [ FD 10 RT 45 ]
```

**NOTE:** If you want to stop the drawing at any point, type CTL-G to stop the action.

#### THE DOT COMMAND

In Aquarius LOGO, the Turtle's trail is not visible when PEN 0 is used. However, the DOT command lets you place a dot on the screen when the Turtle takes its pen up. Type:

```
HOME CS
PEN 0
E 10
DOT
E 10
```

The Turtle moved 10 steps East but didn't leave a trail. On the next instruction it leaves a dot on the graphic screen, then moves on another 10 steps East. Only after the second instruction to move East, does the DOT become visible.

Dots are useful in designing an image of a shape. For example, a regular clock face is less cluttered on the screen, if twelve (12) dots are placed in a circular pattern. To draw the clock face, keep four thoughts in mind:

1. Set the Turtle to PEN Ø (pen up), draw an arc and place a dot on the screen.

2. Repeating the pattern 12 times leaves 12 dots on the screen.

3. Draw a 3Ø degree arc. Type:

```
TO ARC
  REP 1Ø [ FD 1 RT 3 ]
END
```

4. Drawing the ARC 12 times to make a circle ( $12 * 3Ø = 36Ø$  degrees). Instead of leaving a solid line for its trail, use a DOT. Type:

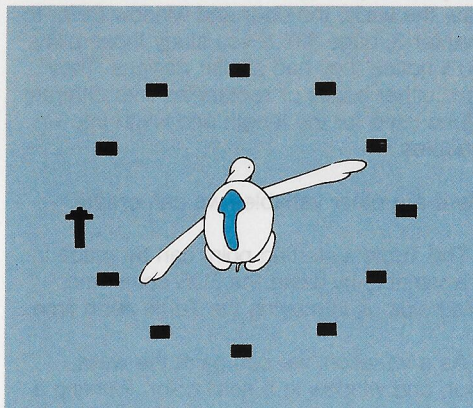
```
TO CIRCLE
  REP 12 [ PEN Ø DOT ARC ]
END
```

The command PEN Ø DOT ARC tells the Turtle to leave a DOT each time it moves through the ARC. Type:

```
HOME CS CIRCLE
```

You may hide the Turtle to show the DOT in the center of the screen by typing:

```
TURTLE Ø
```



## BACK TO THE HOUSE

Multiple variable procedures can save you time in designing your programs. An example of this is the HOUSE program. You recall the many times you typed in each instruction to draw the walls, the door and window (refer to Chapter 3, page 33). If you study those plans, you'll notice they had similar designs. They were either boxes or rectangles with different values used for the length and width into variables.

There are other variables we can create.

1. The Turtle's starting point can be made into a variable by using the POS command. This helps us in moving the Turtle each time.
2. As a variation, we can paint the walls, door, and window in a solid color. Painting a block is created by moving the Turtle forward a certain distance, moving up and back the same amount. Repeating the pattern will give you the width of your block. By making the length and width variables, you can use the

same procedure to draw different parts of the house.

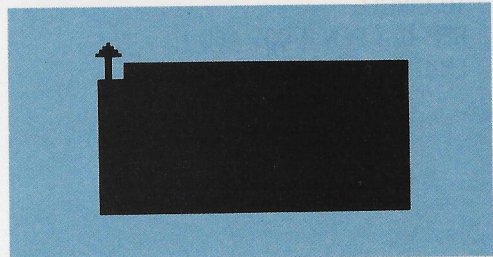
Press **RST** and **RTN** to clear the memory. Type:

```
TO BLOCK :L :H :PLACE  
TILE 255  
PEN Ø  
POS :PLACE  
PEN 1  
REP :H [ E :L N 1 W :L ]  
END
```

Type:

```
TURTLE 1  
HOME CS  
BLOCK 25 15 [ 2Ø 2Ø ]
```

Your screen should look like this:



---

To make a house with a white wall 40 Turtle steps long and 20 Turtle steps high at column 30 and row 56, type:

TO WALL

INK 7

BLOCK 40 20 [ 30 56 ]

END

---

To add a green door, 8 Turtle steps long and 13 Turtle steps high at column 46 and row 56, type:

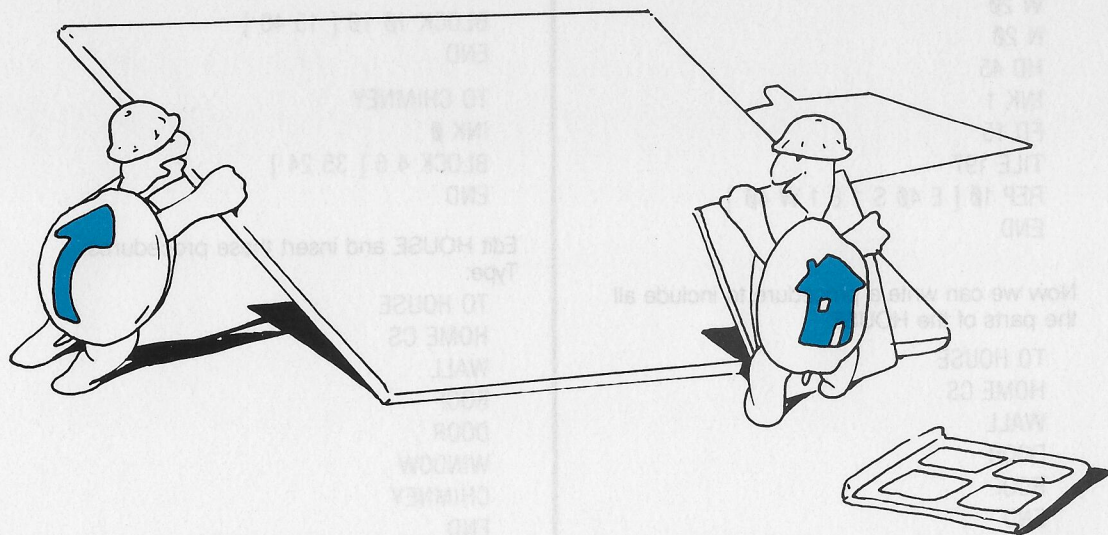
TO DOOR

INK 2

BLOCK 8 13 [ 46 56 ]

END

---



Let's add a side view and the roof to the house in one procedure. The roof is a parallelogram. Use Tile shape 197 for the roof or any other shape from the character list in the Appendix 0.

```
TO ROOF
PEN Ø
POS [ 3Ø 56 ]
PEN 1
INK 7
W 2Ø
N 2Ø
HD 45
INK 1
FD 15
TILE 197
REP 1Ø [ E 4Ø S 1 E 1 W 4Ø ]
END
```

Now we can write a procedure to include all the parts of the HOUSE.

```
TO HOUSE
HOME CS
WALL
DOOR
ROOF
END
```

Type:

## HOUSE

You can add other procedures, such as the window and chimney, however you may need to increase the 'workspace' with additional memory. See Chapter 7 for details. If you have the additional Aquarius memory, here are the remaining procedures.

```
TO WINDOW
INK 4
BLOCK 1Ø 1Ø [ 15 48 ]
END

TO CHIMNEY
INK Ø
BLOCK 4 6 [ 35 24 ]
END
```

Edit HOUSE and insert these procedures.

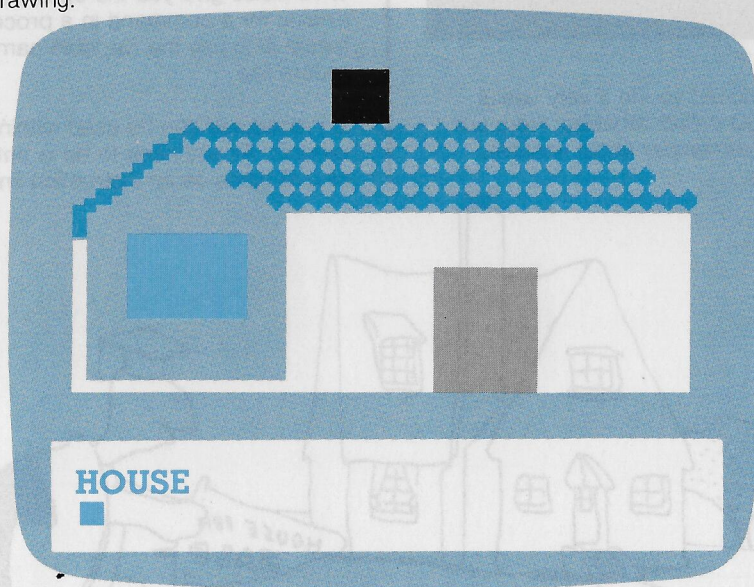
Type:

```
TO HOUSE
HOME CS
WALL
ROOF
DOOR
WINDOW
CHIMNEY
END
```

Type:

HOUSE

to see your drawing.



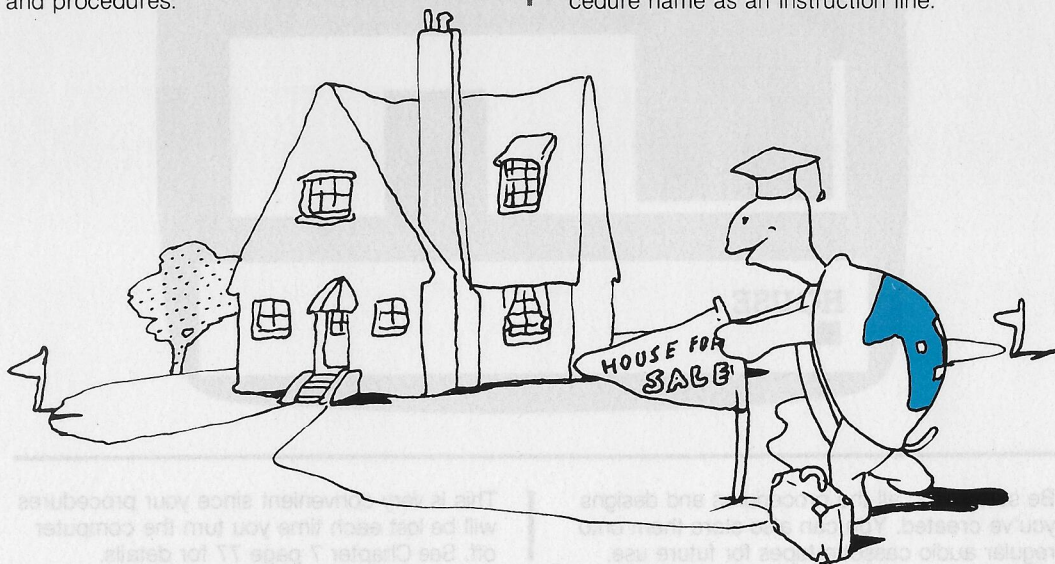
Be sure to log all the procedures and designs you've created. You can also store them onto regular audio cassette tapes for future use.

This is very convenient since your procedures will be lost each time you turn the computer off. See Chapter 7 page 77 for details.

## SUMMARY

Chapter 5 introduced you to a very useful concept in LOGO called variables. Here are some things to remember about variables and procedures:

1. Variables give you the opportunity to enter a value for a command in a procedure. The format is to use the :variable name in a procedure line.
2. Procedures can be used within other procedures. All you need to do is enter the procedure name as an instruction line.



## CHAPTER 6

# ADD LOOPS TO YOUR PROCEDURES

In this chapter we will introduce you to a new set of commands: IFTRUE, IFFALSE, MAKE, and PRINT. These commands add to your LOGO programming skills. You will also learn about recursion. This is the process of adding loops to your procedures.

### LOOPING WITH LOGO

In Chapter 2, you learned about a great time-saver called REPEAT. Now meet its relative, an old-timer that keeps repeating forever until you tell it to stop. This relative is not a com-

mand, but a step in the procedure you write. For example, study this procedure:

```
TO BOX
  REP 4 [ FD 15 RT 90 ]
  BOX
END
```

Once you type BOX, notice that the Turtle repeats BOX over and over again. You can tell it to stop by pressing **CTL-G**. The Turtle responds with the message HALT and stops spinning.

By adding the name of the procedure BOX inside the list of instructions, the Turtle first draws a side by moving forward and making a right turn. Repeating this step four times creates a box. Now the Turtle reads the next instruction, which tells it to make a BOX again, and it repeats the same process. This continuous repetition of the BOX pattern is created by using the BOX instruction inside the BOX procedure. This process is called recursion. Recursion is most helpful when you want to repeat a sequence of steps over and over.

**NOTE:** The recursive instruction is always the last statement in a procedure before you type END.

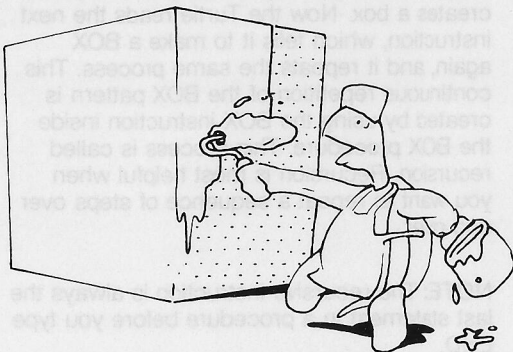
To illustrate, create a rainbow of colors for your BOX procedure. Type:

```
TO COLORBOX
REP 4 [ FD 15 RT 90 ]
INK RN 16 RT 45
COLORBOX
END
```

Now type:

```
COLORBOX
```

You have a colorful box that spins around changing colors as it spins. What happened? You recall that the INK command is followed



by a number. The RN 16 will generate a random number between 0 and 15 (see Reference Section on page 00 for details about the RN command). A random number is drawn for the INK color code. Next, the Turtle turns right 45 degrees and draws a new COLORBOX. The process continues forever because of the recursion statement, COLORBOX.

Try this project and see what a colorful masterpiece you can create:

```
TO SPIRAL :LEN
INK RN 16 PAPER RN 16
FD :LEN RT 90
SPIRAL :LEN + 1
END
```

Now type:

```
HOME CS
SPIRAL 1
```

Notice the + 1 added to the variable :LEN. Each time the FD command in the procedure is called, :LEN increases by 1. A square spiral is drawn. What would happen if the Turtle makes a right turn other than 90 degrees? Let's modify the spiral procedure by making the right turn angle a variable called :ANG.

```
TO SPIRALS :LEN :ANG  
FD :LEN RT :ANG  
SPIRALS :LEN + 1 :ANG  
END
```

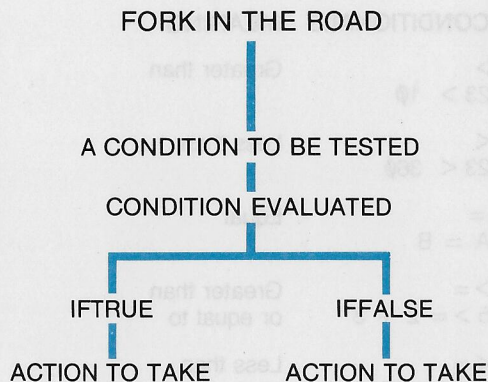
Now type:

```
HOME CS  
SPIRALS 1 89
```

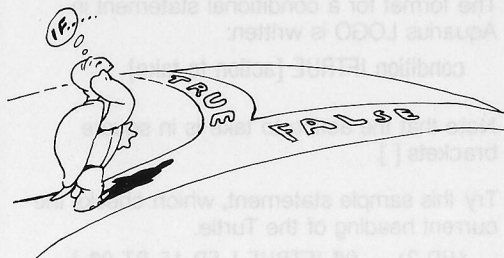
## CONDITIONAL BRANCHING WITH LOGO

In this section, you are going to learn a new set of commands called 'conditionals'. 'Conditional' statements are like forks in a road, and you must decide which way to go. You base your decision on whether some condition exists or not. For example, when you reach the fork, you may wonder whether the bridge ahead is open or closed. If open, you take the right fork. If closed, you take the left.

You use the same thinking with conditional statements in LOGO. You may reach a point in your procedure where you want to test a condition before moving on. If the condition is true, the Turtle takes the route that follows the IFTRUE command. If the condition is false, the Turtle will take the IFFALSE route.



**HELPFUL HINT...**here is a table listing six types of conditional operations found in LOGO. This list will help you develop conditional statements to go along with the IFTRUE and IFFALSE commands.



## CONDITIONALS MEANING

$>$  Greater than

$23 > 10$

$<$  Less than

$23 < 360$

$=$  Equal

$A = B$

$>=$  Greater than

$5 >= 2 + 3$  or equal to

$<=$  Less than

$23 <= 15 + 8$  or equal to

$<>$  Not equal to

$23 <> 15$

## CONDITIONAL STATEMENTS IN LOGO

The format for a conditional statement in Aquarius LOGO is written:

condition IFTRUE [action to take]

Note that the action to take is in square brackets [ ].

Try this sample statement, which checks the current heading of the Turtle.

$(HD ?) = 90$  IFTRUE [ FD 15 RT 90 ]



The statement asks whether the Turtle is currently headed toward the 90 degree position. If this condition is true, move forward 15 Turtle steps and turn right 90 degrees. If your heading is not 90 degrees, LOGO answers with NIL, meaning false and does not move.

Try a similar statement, using the POS command, which includes the option to move the Turtle to a new location if the condition is true.

$(POS ?) = [ 40 30 ]$  IFTRUE [ POS [ 0 0 ] ]

(If the condition is false, LOGO responds with NIL.)

You can create another response using the IFFALSE command:

$(POS ?) = [ 40 30 ]$  IFFALSE [ POS [ 40 30 ] ]

This statement asks if your current position is [ 40 30 ]? If it isn't, move to [ 40 30 ].

## TIME DELAY LOOP

There will be times when you want the Turtle to take a slow walk...take a step, wait a bit, then take another step. This is called a time delay loop. To design a time delay loop, you need two procedures:

1. A count procedure
2. A procedure to move the Turtle forward

Type:

```
TO COUNT :A
:A > 0
IFTRUE [ PRINT :A COUNT :A-1 ]
END
```

```
TO SLOW :A
FD 2 COUNT :A
SLOW :A
END
```

Now type:

```
HOME CS
SLOW 20
```

Slow 20 tells LOGO to start counting backward from 20 to 1, this is accomplished through the recursion COUNT :A-1. Each time it reaches 1, it moves 2 steps forward. Each time the cycle is completed, the numbers are printed in the Command Window from 20 down to 1, (20 counting backwards by 1) LOGO starts the process again with the number you gave it. If you want the Turtle to move faster, then input a smaller number. The lower the number you input, the fewer numbers it must count, the faster the Turtle moves. The conditional :A > 0 establishes the counting limits.

## ANOTHER EXAMPLE OF CONDITIONAL BRANCHING

Conditional branching statements can be used to establish other options within a procedure without having to define another new word. Notice this format uses both IFTRUE and IFFALSE statements.

```
condition IFTRUE [ ACTION ] IFFALSE
[ ACTION ]
```

An example for this format is in the next procedure called MOVE. Type:

```
TO MOVE :LEN  
  :LEN < 10 IFTRUE [ FD 20 ]  
  IFFALSE [ BK 20 ]  
END
```

Type:

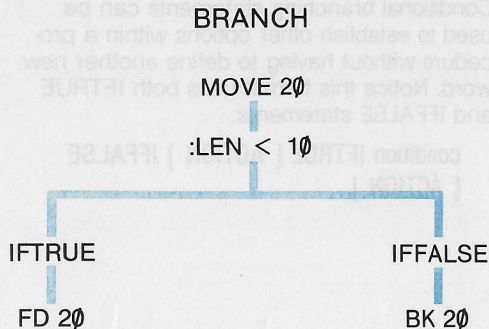
```
MOVE 9
```

LOGO evaluates the input 9, compares it to the set condition :LEN < 10, and concludes that it is true by moving forward 20 Turtle steps. Any number length between 1 and 9 in the command MOVE is accepted as true.

Type:

```
MOVE 20
```

LOGO evaluates the input 20, compares it to the set condition :LEN < 10, and concludes that it is false. The IFFALSE command instructs the Turtle to go back 20 steps. Any number length 10 or over is evaluated in this set of instructions as false.



INPUT 20 FOR :LEN

CONDITION 20 < 10 IS FALSE

BK 20 ACTION IS TAKEN

## SPIRAL FOREVER

By using the IFTRUE and IFFALSE conditional commands, the SPIRAL procedure can operate endlessly without running into the screen boundary. The key is in creating a condition which moves the Turtle HOME by resetting :LEN to a smaller number before the Turtle moves out of bounds. Type:

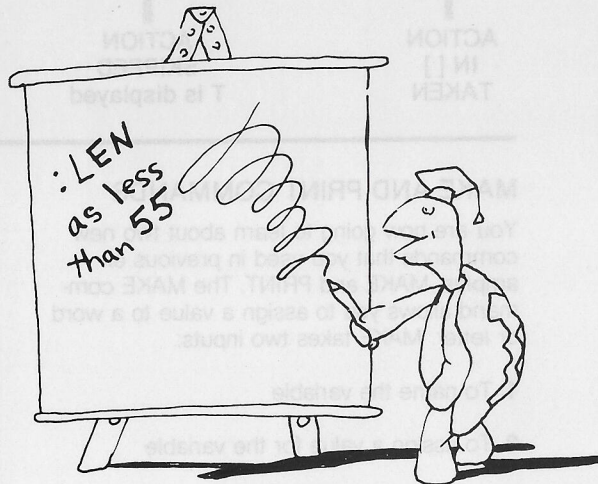
```
TO SPIRAL :LEN
  INK RN 16
  :LEN < 55
  IFTRUE [ FD :LEN RT 90 ]
  IFFALSE [ PEN 0 HOME PEN 1 MAKE
    "LEN 0 HOME CS ]
  SPIRAL :LEN + 1
END
```

You must set the variable :LEN. A condition is established to evaluate :LEN as less than 55. If this condition is true, the Turtle will draw the SPIRAL. Otherwise, IFFALSE, the Turtle is sent back to HOME, and :LEN is set to zero (0) and the procedure begins again. The

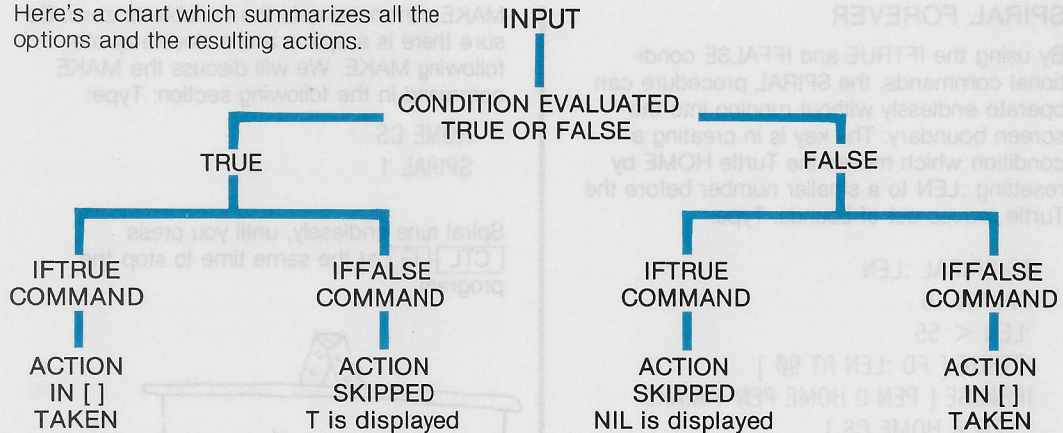
MAKE command sets the variable to zero. Be sure there is a space and a double quote following MAKE. We will discuss the MAKE command in the following section: Type:

```
HOME CS
SPIRAL 1
```

Spiral runs endlessly, until you press **CTL-G** at the same time to stop the program.



Here's a chart which summarizes all the options and the resulting actions.



## MAKE AND PRINT COMMANDS

You are now going to learn about two new commands that you used in previous examples: MAKE and PRINT. The MAKE command allows you to assign a value to a word or letter. MAKE takes two inputs:

1. To name the variable
2. To assign a value for the variable

The format for this command is the following:

**MAKE "idea**      *Assigned value.*

Notice the use of a space and double quote after the command MAKE.

**MAKE "idea 5**      *The word 'idea' now has the value 5.*

**MAKE 'A 20**      *Assigns the value of 20 to A. In LOGO, when assigning values to numbers, only a space is needed.*

**MAKE "grow"big** *Changes the variable 'grow' to 'big'. When assigning a word to another word, LOGO needs another "quote before the assigned word.*

## PRINT COMMAND

To print the value of the two previous examples, type:

**PRINT :A** *Asks for the latest value of variable A. Notice the use of a space and DOTS (:) after PRINT.*

**20** *Latest value for A.*

**PRINT :grow** *Asks for the latest value for the variable grow.*

**big** *Latest value for the variable grow.*

The MAKE command allows you to assign values to letters or words, and convert a name to another name or number. The PRINT command displays the values or words on the screen.

**MAKE "house 5** *Assigns the value of 5 to the variable house.*

**PRINT :house**

*Prints the latest value for the variable house.*

**5**

*Numerical value of the variable house.*

## PUTTING "MAKE" TO WORK BY TAKING A RANDOM WALK

Here's another project using MAKE, PRINT, IFTRUE, and RANDOM. This program is called the RANDOM Walk, where the Turtle's heading and INK color are set to operate randomly. As the Turtle travels, it leaves a colorful trail behind it. With the addition of the Time Delay Procedure COUNT, you can watch the Turtle slowly take its walk.



```
TO COUNT :A
:A > 0
IFTRUE [ PRINT :A COUNT :A-1 ]
END
```

```
TO WALK :A
FD 2
MAKE "B RN 361
HD :B
INK RN 16
COUNT :A
WALK :A
END
```

Type:

```
HOME CS
WALK 15
```

:B is assigned a random variable which ranges from 0 to 360. The random value selected is then assigned to the heading command. As a result, the Turtle moves forward 2 steps, then changes its heading according to the random number assigned.

## MAKING A DIGITAL CLOCK WITH LOGO PROCEDURES

Up to this point, you have used Turtle graphics commands, procedures, recursion and conditional statements. By combining these commands, you can create an interesting project such as putting a 'digital clock' on the screen. You may recall that the clock is similar to the one found on page 11-1 of your "AQUARIUS GUIDE TO HOME COMPUTING."

### COUNTING LIKE A CLOCK

In the Time Delay Procedure, you taught LOGO to count backwards, from 20 to 1 using an IFTRUE statement with a counter :A-1. What about teaching LOGO to count forward like a clock, from zero to 60? A clock has three time displays, one for the hour, one for the minute, and one to display the seconds. To create our clock, we will just work with the minute and seconds display first.

Since 60 seconds equal 1 minute, our first job is to build a timer that begins with zero and ends at 60. Once that timer reaches 60 it adds a + 1 to the minute side of the timer, and returns the seconds side of the timer to

zero. To keep track of the variables we are using to build our clock, look at the charts below.

## VARIABLE NAME REPRESENTS

:M	Minute digits
:S	Seconds digits

The following LOGO commands are used:

COMMAND	REASON
PEN Ø	No Turtle tracks needed
TURTLE Ø	Makes Turtle invisible
> =	Evaluates condition
IFTRUE	Conditional statement
TP	Turtle message displays the digital time

The clock is set at the beginning of the procedure with variables :M and :S input by you. A counting instruction increases the seconds by one infinitely. For example:

```
TIME :M :S + 1
```

A conditional branch increments the minute timer by one each time the seconds timer reaches 60. Then it returns the seconds display to zero. This is done by using an IF-TRUE statement.

```
:S >= 60 IFTRUE [ MAKE "M :M + 1
MAKE "S 0 ]
```

Finally, since a digital clock displays two sets of numbers, one for the minute display, and one for the seconds display, two Turtle messages are needed.

Let's put it all together. Type:

```
TO TIME :M :S
PEN Ø TURTLE Ø
:S >= 60
IFTRUE [ MAKE "M :M + 1 MAKE "S 0
]
:M >= 60
IFTRUE [ MAKE "M 0 ]
HOME
CTP :M
POS [ 48 30 ]
CTP :S
TIME :M :S + 1
END
```

Wait a minute! What is CTP? CTP is the name of a procedure to run the time display properly. If you run the procedure with just TP, something funny will happen to the numbers displayed. They will change erratically in the time display. The Turtle leaves a trace of the second digit when displaying a single digit number at the Turtle position. What we need is a program to eliminate this 'trace' business, and print a place holder 'Ø' when numbers less than 1Ø are displayed. Type:

```
TO CTP :NUM
:NUM < 1Ø
IFTRUE [ TP Ø E 2 ]
TP :NUM
END
```

If the number is less than 1Ø we ask the Turtle to print the number Ø then move East 2 steps before printing the single digit (Ø to 9). To see how CTP works, type:

```
TURTLE Ø
CTP 55
CTP 9
```

First you have a two digit number 55. The second time, Ø9 is displayed. Are you ready for the time display? Type:

## TIME 1Ø 1Ø

You now have Aquarius LOGO counting like a clock. When you want to stop the clock, type:

## CTL-G

**HELPFUL HINTS:** It may appear confusing that the variable NUM in procedure CTP is changed to :M or :S when CTP is included in the TIME procedure. What happens is what we call 'VALUE PASSING' and the variable NUM is used as a 'Dummy Variable'.

In programming, there are times when 'dummy variables' are created, waiting for a value from one procedure to take its place from a previous statement. A good example of a 'dummy variable' is the use of a 'joker' in a card game. The 'joker' takes the place of another card. All it requires is a definition by the player. The same process takes place with 'dummy variables', another variable can take its place as soon as it is defined by the user. Dummy variables add flexibility to your Logo procedures.

Due to the complexity of the Digital Clock procedures, additional memory will be needed to complete the program. See chapter seven about expanding the work space.

Creating a digital clock takes three steps:

1. Place the shape of a clock on the screen, using 12 dots to form a circle.
2. Place the word "TIME" on the screen and the symbol ":" for the time display in the middle of the clock face.
3. Display the current time on the clock.

Now begin making the shape of a clock:

```
TO START
PEN Ø
POS [ 4Ø Ø ]
HD Ø
TURTLE Ø
END
```

This procedure places the Turtle in the proper position to start a circle.

```
TO ARC
REP 15 [ FD 1 RT 2 ]
END
```

ARC tells the Turtle to make an arc of 3Ø degrees. The Turtle moves forward 1 unit, turns right 2 degrees, 15 times.

```
TO CIRCLE
REP 12 [ PEN 1 DOT PEN Ø ARC ]
POS [ 32 26 ]
TP 'T I M E'
POS [ 37 3Ø ]
TP ':'
END
```

This makes a circle using the ARC procedure. 12 ARCs form a 36Ø degree circle. Each point in the ARC is marked by a dot. Next move the Turtle to position 32 36 to display the word 'TIME'. Move the Turtle to print the time display using ':'. There are 4 spaces between the colons.

To check your procedures, type:

```
START CIRCLE
```

Small black dots appear to make the circle. Then 'TIME' and the ':' are displayed on the clock face. At this point, you can change the color and the shape of the dots by using the INK and TILE commands. For example, type:

```
INK 5 TILE 128
START CIRCLE
```

The clock face has a violet colored shape. The shape is character 128. By referring to the color table found on page 63 and the character list on page 00 you can select any shape and color you desire.

## CREATING THE DIGITAL TIME DISPLAY

Before you proceed, you need to be aware that the following "DGTIME" procedure requires extra memory. You can increase the memory of Aquarius by referring to page 77 in Chapter 7.

Now create the time display:

```
TO DGTIME :H :M :S
:S >= 60
IFTRUE [ MAKE "M :M + 1 MAKE "S 0 ]
:M >= 60
IFTRUE [ MAKE "H :H + 1 MAKE "M 0 ]
:H >= 13
IFTRUE [ MAKE "H 1 ]
COUNT 70
HOME
CTP :M
POS [ 48 30 ]
CTP :S
POS [ 32 30 ]
```

```
CTP :H
DGTIME :H :M :S + 1
END
```

The DGTIME procedure should look familiar to you. The same principles used to write TIME in the previous section are used for this time display. The only difference is the addition of the hour display :H. In fact, many of the procedures we are using, such as ARC, CIRCLE, COUNT, and CTP were presented earlier.

For DGTIME to operate without an error message such as, "I DON'T KNOW HOW TO DGTIME", be sure all the procedures have been entered in the 'workspace'. To complete the Digital Clock, we will add the COUNT and CTP procedures. Type:

```
TO COUNT :A
:A > 0
IFTRUE [ COUNT :A-1 ]
END
```

```
TO CTP :NUM
:NUM < 10
IFTRUE [ TP 0 E 2 ]
TP :NUM
END
```

To recall the discussion for these two programs, review the previous section on COUNTING LIKE A CLOCK and TAKING A RANDOM WALK.

## COMPLETING THE CLOCK COMMAND

### Reminders:

1. LOGO is very particular when asked to work with negative numbers. In your TO COUNT procedure, IFTRUE [ COUNT :A-1 ], :A-1 can be input :A-1 or :A - 1 but not :A -1. Otherwise the program will not run, and an Error Message will say "No room."

2. A good way to determine whether you have any 'bugs' in your program is to run each procedure separately, or with its linking procedure. For example, earlier you were asked to run a check using the START and CIRCLE procedure. This helps to catch 'bugs' before they become difficult to find.

### COMPLETING THE CLOCK

In building your 'digital clock', you now have in place six procedures: START, ARC,

CIRCLE, COUNT, CTP and DGTIME. You need to link all the procedures together to operate under one command, CLOCK.

```
TO CLOCK :H :M :S
START
CIRCLE
DGTIME :H :M :S
END
```

CLOCK ties all the procedures together. Using a single command and three values for hour, minutes, and seconds, type:

```
CLOCK 1 20 30
```

Use your current time to set your clock. Remember, you can change the color of your clock by using the INK command, and the TILE command to change the shape of the face.

## CHAPTER REVIEW

Chapter 6 introduced many new concepts: recursion, conditional branching such as IFTRUE, IFFALSE, along with two commands, MAKE and PRINT. Listed below is a quick summary of these commands:

- |                                 |   |
|---------------------------------|---|
| <b>1. Recursion</b>             | Creates loops within a procedure by calling upon its own name.                |
| <b>2. Condition IFTRUE [ ]</b>  | Tests a condition and indicates the action to take if the condition is true.  |
| <b>3. Condition IFFALSE [ ]</b> | Tests a condition and indicates the action to take if the condition is false. |
| <b>4. MAKE "idea value</b>      | Assigns values to a variable. Takes two inputs, idea and assigned value.      |

### 5. PRINT

Tells LOGO to print a number, or word on the text screen.

### 6. RN number

Tells the computer to select a random number from zero to one less than the number you select.

### 7. Conditional Operators

>	Greater than
<	Less than
=	Equal to
> =	Greater than or equal
< =	Less than or equal
< >	Not equal to

## CHAPTER 7

# MANAGING YOUR AQUARIUS LOGO WORKSPACE & STORAGE

In your Aquarius LOGO, all procedures and variables are temporarily stored in the computer's memory using RAM (random access memory) or the 'workspace'. Each time a procedure is programmed, it occupies a part of the workspace. LOGO temporarily remembers all procedures as long as the computer is ON. Once the computer is turned OFF, it forgets what is in its memory.

Whenever you see the message that reads "NO ROOM", LOGO is saying that you have filled the memory or 'workspace' with all the procedures you've typed in. Here are four things you can do when this happens:

1. Reorganize the workspace using the ERASE and NAMES commands.
2. Add an AQUARIUS™ Mini Expander with 4K or 16K of additional memory.
3. Save 'current memory' on a cassette tape using an Aquarius Data Recorder.
4. Empty 'current memory' by pressing the **RST** key on the upper left corner of the keyboard. By taking this action, you restart LOGO again with an empty workspace.

Chapter 7 discusses how you can manage your Aquarius workspace using any one of these options.

### REORGANIZE THE WORKSPACE USING 'ERASE' AND 'NAMES' COMMANDS

To recall which programs currently occupy the workspace, the NAMES command in LOGO allows you to review the names of the procedures and variables there. Type:

**NAMES**

LOGO responds with a list of the names you've used and defined. [name1 name2 name3 name4 ... studs]

Studs is the name for the Turtle, that is, the one in the HOME position when you first turn LOGO on. Since the Aquarius LOGO has multiple Turtles (discussed in Chapter 8), the main Turtle is called Studs. As you create other Turtles, you may name them as you wish.

Once you've reviewed the procedures in the workspace, you may erase any procedures you no longer need. To do this, use the ER (short for ERASE) command. As an example, you may no longer need the SPIRAL program. Type:

**ER "SPIRAL**

LOGO responds with the name of the procedure SPIRAL. Once the procedure is erased, LOGO is ready to proceed with your next command. To check whether the program has been erased, type:

**SPIRAL**

LOGO responds with:

**"I DON'T KNOW HOW TO SPIRAL"**

You now have some additional workspace.

Here's two helpful hints for you to consider:

1. Use the P "procedure name to review the unwanted procedure before using the ER command. This minimizes the risk of mistakenly erasing a program you want to keep.
2. Use NAMES periodically to review the names that have been used. This avoids renaming a procedure by writing over it with a new procedure.

## **AQUARIUS™ MINI EXPANDER**

If your current workspace is insufficient after using the ER command, you can increase the workspace by installing the Aquarius Mini Expander and Aquarius Memory cartridge. You have a choice between additional 4K or 16K memory to add.

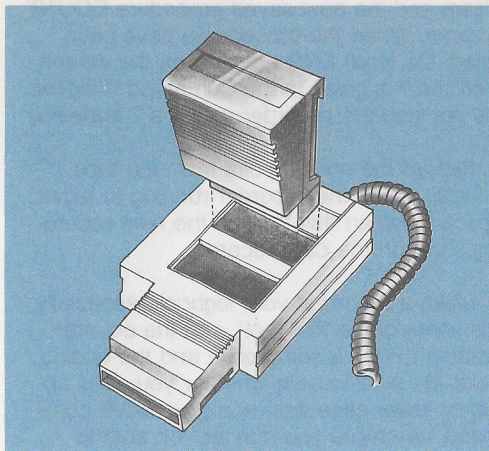
The Aquarius Mini Expander can be plugged directly into the cartridge port of your computer. It comes with two ports. By plugging the LOGO cartridge into the front port marked

“Program”, and the memory cartridge into the back port marked “Memory”, you have expanded the Aquarius LOGO workspace.



#### REMINDER:

Turn the computer off when inserting and removing the Mini Expander and the LOGO cartridge. You will lose whatever is in the computer memory.



Using the Aquarius Mini Expander & either 4K or 16K memory cartridge to create additional workspace.

\*Sold separately

## SAVING ‘LOGO’ PROGRAMS ON THE AQUARIUS DATA RECORDER

Information in the ‘workspace’ is lost once the computer is turned OFF. Instead of typing in your procedures each time you use the computer, you can save them on cassette tape by using the Aquarius Data Recorder.

There are three commands in LOGO for using the data recorder: **SAVE**, **LOAD**, and **CHECK**.

**SAVE** tells Aquarius to save your procedures on a cassette tape.

**LOAD** tells Aquarius to load the programs from your cassette back into the workspace.

**CHECK** asks Aquarius to verify whether the programs have been correctly saved onto the cassette recorder.

## SAVING ‘LOGO’ PROGRAMS WITH ‘SAVE’, ‘LOAD’, AND ‘CHECK’

### Using SAVE

To save your LOGO programs, follow the next nine steps:

1. Connect the Aquarius Data Recorder to the Aquarius computer as described in the data recorder manual.

2. Rewind your tape recorder. Set the tape counter to "000". Forward the tape past the tape lead. Record the counter reading as your loading point, or forward the tape to the next available space as noted in your logbook.

3. Select a 'filename' to identify the set of procedures you are saving. For example, you can call them TEST. In reviewing your NAMES, save the WALK and COUNT procedures discussed in Chapter 6.

4. Type: **SAVE 'test' [ WALK COUNT ]**

**DO NOT PRESS [RTN] KEY YET!!!**



**REMINDER...**

- 1) Notice a space between SAVE and the single quote around 'test'.
- 2) Indicate in your logbook whether the filename is upper or lower case letters.
- 3) Do not press the [RTN] key.

5. Press the [RECORD] and [PLAY] keys simultaneously on the data recorder.

6. Now press the [RTN] key on the computer keyboard. This prevents you from losing any of your program before you save it.

7. As the procedures are recorded, they are displayed on the screen with a list of all the commands. For example, each step in the WALK procedure is listed. Then the screen clears and the next saved procedure COUNT is given. When the procedures have been listed, the word END appears on the screen, then the ? is given in the Command Window. Do not stop the recorder until the ? appears.

8. Record the ending point from the tape counter. Keep the recorder running to leave space between this file and the next recording. Turn off the data recorder.

9. Make a record in your logbook, especially the name of the file 'test', and the counter reading for the loading point and the ending point. Be sure to note whether you used upper or lower case letters in your SAVE command, otherwise you may not be able to recall your procedures when you reload.

```

TO WALK :A
  FD 2 MAKE "B RN 361 HD :B INK RN 16
COUNT :A WALK :A
END

```

Here are some other helpful hints you might want to keep in mind:

1. You may save as many procedures as you wish under one filename. Simply place them inside the square brackets separating each by a space. For example, type:

```

SAVE 'test' [name1 name2 name3
name4 ... ]

```

2. If you wish to save all the procedures in your workspace under a single filename, you can use the NAMES command instead of the [ ]. For example, type:

## SAVE 'test' NAMES

Note that brackets are not used.

3. Sometimes, you may accidentally lock up the system during the SAVE operation. First, stop the recorder. Then you may 'warm start' the system without erasing the workspace. Just press **RST**, then press **CTL C** (CTL key and letter C key at the same time). You will be back in the command mode to restart the SAVE operation again. Be sure it is the **CTL C** keys, and not the **RTN** key. The latter will empty your workspace.

## USING 'CHECK'

The CHECK command is used to verify a saved file. For example, you saved 'test'. To determine whether it has been properly saved on the data recorder, follow the next five steps:

1. Check your equipment to be sure all connections between the Aquarius and the data recorder are in place.
2. Rewind the tape to a number just before the loading point you've noted in your logbook. For example, the starting point for the file 'test' is "0000".

---

**3.** Type CHECK 'filename' [procedures saved under the filename] which is CHECK 'test' [WALK COUNT] in this case. Be sure there is a space between CHECK and 'test'. For example, CHECK 'test' [WALK COUNT].

The procedure must also be in the exact sequence as the SAVE command you type.

**4.** Press the **RTN** key on the keyboard, then the **PLAY** key on the data recorder.

**5.** LOGO verifies the information on the tape against the information in workspace. If they match, LOGO automatically displays verified procedures and returns a ? You will be in the Command mode to continue your programming. If the filename does not match, "#" appears. A # sign is displayed each time a file or a procedure is skipped. If the procedures are not properly saved, an error message says "I DON'T UNDERSTAND 40". If this occurs, rewind the tape and resave your files.

## USING 'LOAD'

The steps to LOAD your programs from the data recorder are the same as those you used to CHECK your files, except the command LOAD is used. There is no need to list your procedures.

**1.** Check the connections between your equipment.

**2.** Rewind the tape to a number just before the starting point you've noted in your logbook. For example, the starting point for the file 'test' is "000".

**3.** Type LOAD 'test'. Note the space between LOAD and 'test'. Check your logbook to be sure you are using upper or lower case letters or both. LOGO distinguishes between upper and lower case in filenames. Otherwise LOGO will not be able to identify the file you intend to LOAD.

**4.** Press the **RTN** key on the keyboard, then the **PLAY** key on the data recorder.

**5.** LOGO reads your files and displays the text of the procedures on the screen. LOGO will automatically skip the files until the file you want is found. A # sign is displayed each time a file or a procedure is skipped.

**6.** Once the files are entered into memory, a ? appears in the text window and you are ready to start programming again.

**NOTE:** If you stop the recorder before the ? appears, the system locks up and you are not able to start programming. If this occurs, press **RST** and **CTL** **C** to start your LOAD procedures again.

## HELPFUL HINTS

1. Remember to press the **RTN** key before the starting point is passed.

2. A shortcut...instead of typing in the filename, use two single quotes after the command. For example, type:

**LOAD ' 'RTN**

This tells the data recorder to load the first file it meets. Remember to leave a space between the command and the single quotes.

## USING THE AQUARIUS™ LINE PRINTER

A useful way to keep a file of your LOGO procedures is to keep a printout of the procedure. Put them in your logbook. It also helps to have a 'hard copy' for analyzing the 'bugs' in your LOGO program. The command to activate your Aquarius Printer is LP (short for line printer). Follow the next six steps to obtain a copy of your program:

1. Connect the AQUARIUS™ Printer as described in the Aquarius Printer manual.

2. Switch the printer to the 'Text' mode located on the back of the printer.

3. Type LP followed by a space, double quotes, and the procedure name. For example:

**LP "WALK**

4. The Aquarius Printer types out the WALK procedure.

5. For a series of procedures, you can use the following format:

**LP [name1 name2 name3 ...]**

For example, LP [ COUNT WALK ] gives you the text of the two procedures: COUNT and WALK.

6. To print all the procedures found in the workspace, type LP and the NAMES command.

**LP NAMES**

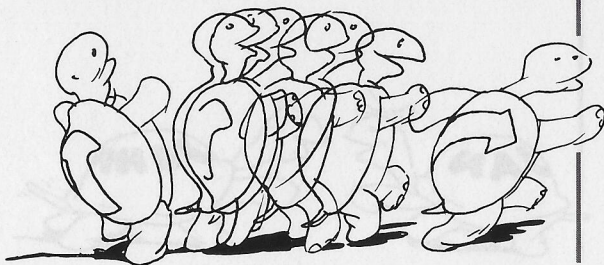
You now have a copy of all the procedures in the workspace.

## REVIEW MANAGING THE AQUARIUS WORKSPACE

COMMANDS	DESIRED ACTION
ER "procedure or variable name"	Erases procedure and variable names found in the workspace
NAMES	Lists all the procedures and variable names found in the workspace.
SAVE 'filename' [ name1 name2... ]	Saves desired procedures from workspace to data recorder
CHECK 'filename'	Verifies saved files with data in workspace
LOAD 'filename'	Procedure that loads files from data recorder to workspace
LP "filename"	Command that uses the Aquarius Printer for hard copy of procedure

## CHAPTER 8

# MULTIPLE TURTLES AND ANIMATION



Chapter 8 introduces you to the use of Multiple Turtles. Commanding more than one turtle allows you to create a variety of scenes with animation. The key is having the Turtles work together. To help you with this process, you'll be introduced to four new commands: HATCH, ASK, DELAY, and SHAPE.

### CREATING MULTIPLE TURTLES

When you create additional Turtles, you can operate each independently or they can work cooperatively in pairs or groups. This is done by giving commands to each Turtle. To begin, we need to learn how to 'hatch' new Turtles, and how to speak to each one.

When you turn on Aquarius LOGO, the first Turtle you work with is named 'Studs'. In the previous chapters, you've been working with Studs. Now let's create a teammate for Studs. The command that creates another Turtle is HATCH 'name of new Turtle. Let's call the new Turtle Jan.

Type:

**HATCH "JAN**

LOGO responds with the name of the new Turtle, JAN. Notice however, that Jan is not visible. Jan is hiding behind Studs. She also

looks like Studs. To show the two Turtles on the screen, you'll need to ask them to move. Move Studs forward by typing `FD 10`. Type `CS` and you will see Jan at HOME. To move Jan, use another new command, `ASK`. The format is `ASK :name of Turtle`.

Type:

```
ASK :JAN HD 180 FD 10
```

JAN moved West. The `ASK` command will only send a message to the Turtle named and no other. Each Turtle must be addressed separately.

In Chapter 10 we will introduce a procedure to speak to all the `HATCHed` Turtles using one set of instructions.

If you type `CS`, both Turtles will remain on the screen. There are times when you have multiple Turtles on the screen, and you may forget which Turtle you are addressing. To help you out, type:

```
ASK ?
```

The name of the current Turtle will come on the text screen.

Let's create another Turtle named John.  
Type:

```
HATCH "JOHN
```

Where is John? John is hiding behind Jan. Any newly created Turtle will take on the characteristics of the last Turtle. This means that John will carry the same shape as Jan as well as any other properties, such as: position, heading, ink, visible or invisible, pen up or down, and tile. In other words, each newly created Turtle is the 'clone' of the previous Turtle.

To make John visible, give him a new position and type `CS`. Type:

```
ASK :JOHN POS [ 10 55 ] CS
```



## PUTTING THE TURTLE TO WORK

Multiple Turtles can add dramatic effects to your programs. At the same time they reduce the programming burden created when you work with only one Turtle. To demonstrate, we'll use three Turtles to create three snowflakes that continually change colors. First we'll hatch two additional Turtles and position them on the screen with Studs (using the TURTLES procedure). Next we'll create a box with random colors. The COLORBOX procedure draws a box and then rotates the Turtle heading by 45 degrees. This allows each Turtle to create an endless pattern of changing colors with a recursive call to the procedure SPINBOXES.

Press **RST** and **RTN** to clear the workspace.

Type:

```
TO TURTLES
PEN Ø
HATCH "JAN
```

```
HATCH "JOHN
ASK :STUDS
POS [ 4Ø 15 ]
PEN 1
ASK :JAN
POS [ 6Ø 45 ]
PEN 1
ASK :JOHN
POS [ 2Ø 45 ]
PEN 1
END
```

```
TO COLORBOX
INK RN 16
REP 4 [ FD 1Ø RT 9Ø ]
RT 45
END
```

```
TO SPINBOXES
ASK :STUDS
COLORBOX
ASK :JAN
COLORBOX
ASK :JOHN
COLORBOX
SPINBOXES
END
```

Type:

## TURTLES

to hatch Jan and John. Once they are hatched, type:

## CS SPINBOXES

Press **CTL** - **G** to stop the program.

**HELPFUL HINT:** As new Turtles are HATCHed, remember they take on the characteristics of the current Turtle. Notice in TURTLES, we only used PEN 0 once. Each new Turtle was also in a PEN UP state. In speaking to each Turtle after new turtles are HATCHed, always move the current Turtle first. Then move each new Turtle in the order they are HATCHed. For example, in SPINBOXES, Studs was moved first, then Jan and John. If the sequence is not followed, an image of the off-sequence Turtle may be left behind in the original Turtle position. In other words, its shadow may remain giving the impression that another Turtle was HATCHed. To clear the confusion type CS, and only the HATCHed Turtles will remain on the screen.

**NOTE:** See Reference Section for discussion on Turtle shadows.

## TURTLE SHAPES AND ANIMATION: A BOUNCING BALL

Animated graphics can make programming in Aquarius LOGO fun and exciting. Moving objects can be created that resemble a bouncing ball, a flying airplane or a running man. By using the SHAPE command, you can use the code number found in the Character Set, page 00 to change the Turtle's shape. Don't confuse the SHAPE command with the TILE command. The former changes the Turtle's appearance, while the latter changes the Turtle's track.

To create two bouncing balls, one ball to bounce horizontally, with the other bouncing vertically, two procedures are needed. The first procedure will HATCH another Turtle. It will also include instructions to assign a SHAPE, INK, and POS for both Turtles. The second procedure will ASK each Turtle to move first in one direction, then reverse its direction a set number of times back and forth. By making the procedure recursive, the balls will bounce forever.

Press **RST** and **RTN** to clear the workspace. Type:

```
TO SETUP
PEN 0
SHAPE 150
HATCH "JOHN
ASK :STUDS
POS [ 40 55 ]
INK 1
ASK :JOHN
POS [ 20 30 ]
INK 4
END
```

To see the red and blue balls, type:

**SETUP**

In animation, the sensation of movement is created by moving objects in very small steps. REP 50 moves the ball many times in each direction to create the feeling of motion.

To stop the program, type CTL-G. Type:

```
TO BOUNCE
REP 50 [ ASK :STUDS N 1 ASK :JOHN E 1 ]
REP 50 [ ASK :STUDS S 1 ASK :JOHN W 1 ]
BOUNCE
END
```

Type:

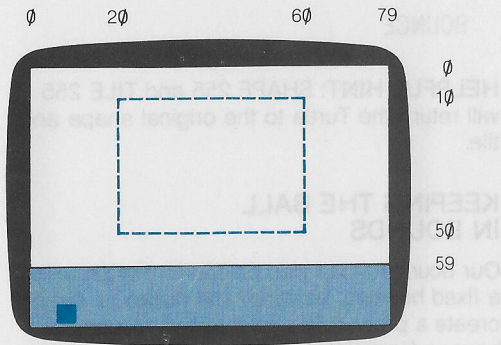
**BOUNCE**

**HELPFUL HINT:** SHAPE 255 and TILE 255 will return the Turtle to the original shape and tile.

## KEEPING THE BALL IN BOUNDS

Our bouncing ball was placed in motion using a fixed heading, position, and distance. Let's create a procedure which allows the ball to bounce freely in random motion without hitting the screen boundary and ending the program.

To do this, we will need to establish a fence for the ball. The playground fence is defined by the screen columns and rows. Each time the ball hits the edge of the fenced-in playground, it will bounce back and continue its path in another direction. This means it must be given a new heading. To establish our boundary, we need to identify the borders of the playground. The playground will fall between columns 20 and 60, and between rows 10 and 50. (See page 90)



#### NEW PLAYGROUND FOR THE BOUNCING BALL

A procedure must be written to identify the current location of the ball as it bounces inside its playground. Two variables are created to identify the current position: position X, the column number, and position Y, the row number. They'll be identified as PX and PY.

Since the POS ? returns a list of the current position by column and row number, we need to identify each number separately. Two new commands are used, FIRST, and LAST. (Both are discussed in detail in Chapter 10.) FIRST will give us the first number, the column location in the POS command. This will be the PX variable. LAST will give us the second

number, the row location in the POS command. This will be the PY variable. For example, if the Turtle is located in POS [ 20 10 ], 20 will be assigned to PX, and 10 will be assigned to PY. Press **RST** and **RTN** to clear the workspace, then type:

```
TO PXY
  MAKE "PX FIRST (POS ?)
  MAKE "PY LAST (POS ?)
END
```

The boundary conditions for the fence are established in the next procedure, with instructions to move in the opposite direction when the ball hits the barrier. To identify the current location of the ball, the procedure PXY is included. Type:

```
TO PLAYGROUND
  PXY
  :PX <= 20
  IFTRUE [ HD 0 DOT ]
  :PX >= 60
  IFTRUE [ HD 180 DOT ]
  :PY <= 10
  IFTRUE [ HD 270 DOT ]
  :PY >= 50
  IFTRUE [ HD 90 DOT ]
END
```

The following summarizes the boundary conditions in PLAYGROUND.

PLAYGROUND BOUNDARY	NEW DIRECTION
Left wall :PX <= 20	HD 0
Right wall :PX >= 60	HD 180
Top wall :PY <= 10	HD 270
Bottom wall :PY >= 50	HD 90

As each wall is hit, a DOT is left behind to indicate the borders of the playground.

To place the Turtle in random motion within the playground, the instruction [ PLAYGROUND FD (RN 5) ] is used with a REP command. Remember REP gives an animated effect to the ball. Random colors and headings are included. Type:

```
TO BOUNCE
  INK RN 16
  REP 10 [ PLAYGROUND FD (RN 5) ]
  HD RN 361
  BOUNCE
END
```

To set the ball in motion, place STUDS in a PEN UP state, and change its shape to a ball. Type:

```
HOME CS
PEN 0
SHAPE 150
TILE 128
BOUNCE
```

What you see is a bouncing ball with a life of its own, running into an invisible barrier and bouncing back. Each time it hits the fence, a colorful square TILE 128 appears. Sometimes the ball will move faster, sometimes slower, depending upon the effect produced by forwarding the Turtle a random number of steps when the FD (RN 5) instruction is encountered. The playground slowly becomes visible as the ball bounces around the screen.

## MAKING AN ANIMATED FIGURE WITH TWO TURTLES

Up to now, our discussion of multiple Turtles focused on assigning each Turtle a specific job. We will now HATCH two Turtles next to each other and having them work cooperatively as one. Our next project places two Turtles together to create a running man on your screen.

Three steps are used to design a running man.

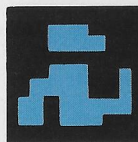
1. Hatch two Turtles, one beneath the other.
2. Change the Turtle shapes to represent the upper and lower halves of a man.
3. Move the two Turtles as one to animate the running figure.

Before you proceed, press **RST** and **RTN** to clear the workspace. You can save your programs on a cassette tape as described in Chapter 7.

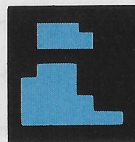
## FINDING THE CORRECT SHAPE

You now have an empty workspace with STUDS on the screen. To change STUDS' shape, type:

**SHAPE 23**



23



21

Studs has the shape of a man, namely his upper half with extended arms.

Typing SHAPE 21 shows the man in another position. The man's arms are now in a down position. By switching from one shape to another, we can create a sensation of movement. Type:

**REP 20 [ SHAPE 23 DELAY SHAPE 21 DELAY ]**

Notice the DELAY command. The command introduces a pause between the changing SHAPE commands. This slows the time between shapes to enable the eyes to see the changing movement. Without the DELAY command, the computer moves the shapes so rapidly that a blur results.

## HATCHING TWO TURTLES TO MAKE A RUNNING MAN

Along with STUDS, another Turtle is HATCHED, and placed below STUDS. Type:

```
TO SETUP
ASK :STUDS
PEN Ø
POS [ 1Ø 3Ø ]
HATCH "HEAD
S 3
END
```

STUDS is placed in a PEN UP state. A new Turtle named HEAD is hatched inheriting the PEN UP state from STUDS. STUDS is moved 3 steps South, underneath HEAD. STUDS will become the lower half for the man. Type:

```
SETUP CS
```

Two Turtles with an arrow shape are displayed on the left side of the screen. The one on top is the hatched Turtle named HEAD, the one below is STUDS.

Next we will assign the shapes to the Turtle. Since there are four shapes, two for the upper half and two for the lower half, a variable must be used to accept each of the different shapes as the man moves forward. The variables SH represents the shapes for the upper half and SF represents the shapes for the lower half. A variable, DGS for degrees, is used for the heading to let the man move in an East or West direction. Type:

```
TO MOVE :SH :SF :DGS
ASK :HEAD
INK 2
SHAPE :SH
HD :DGS
FD 1
ASK :STUDS
INK 1
SHAPE :SF
HD :DGS
FD 1
DELAY
END
```

Both HEAD and STUDS are asked to move the same distance. This keeps the figures together, otherwise they will split apart.

Since :SH :SF and :DGS will require multiple inputs for each, the values will be assigned by the next procedure. Type:

TO RUNMAN

```
REP 25 [ MOVE 23 24 Ø MOVE 21 22 Ø ]
```

```
REP 25 [ MOVE 13Ø 131 18Ø MOVE 14Ø 141 18Ø ]
```

RUNMAN

END

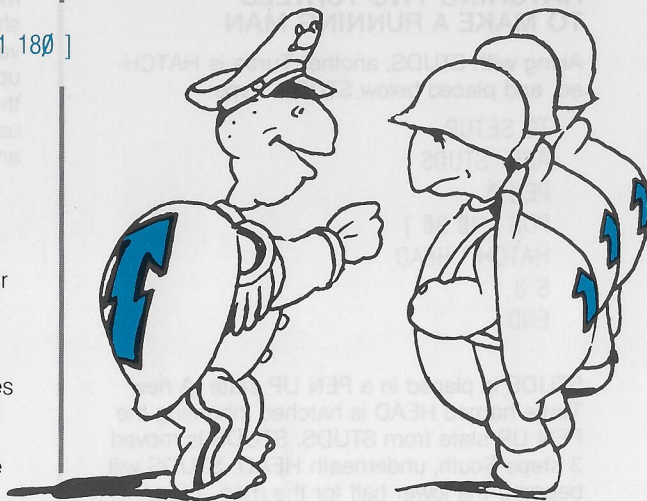
To start the man running, type:

**RUNMAN**

A man with a green upper half and red lower half will appear on the screen and run back and forth.

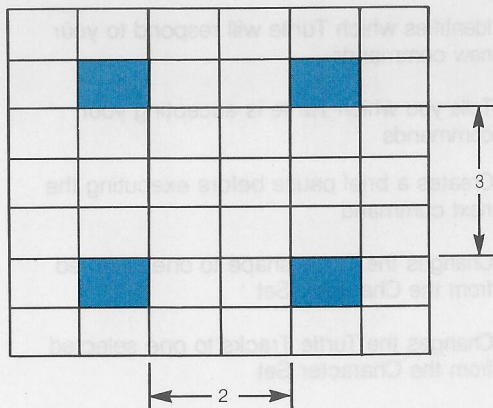
In Aquarius LOGO, the character set includes portions of a running man figure. Characters numbered 21, 22, 23, and 24 are the elements of a running man facing the right side of the screen, and 13Ø, 131, 14Ø, and 141 are of a running man facing the left side. We will use all eight shapes in this program.

## REVIEW MULTIPLE TURTLES AND ANIMATION



Listed below are some things to remember when designing procedures using multiple Turtles and animation.

1. Determine how many Turtles you will need and what character shapes you will give them.
2. Decide how to position the Turtles next to each other. Allow three units for a vertical placement of the next Turtle, and if you are thinking of a horizontal placement, use two units.



3. Hatch and name each Turtle for the symbol it represents.

4. Move the Turtles in the same procedure with similar headings and penstates. Include a DELAY command to allow the animation to appear.

5. Decide whether you want to leave Turtle Tracks, if not use PEN Ø.

6. If there are extra Turtles on the screen you may want to hide them, or have them create other objects on the screen.

7. Create one procedure to link all the other procedures together.

## SUMMARY OF MULTIPLE TURTLE AND ANIMATION COMMANDS

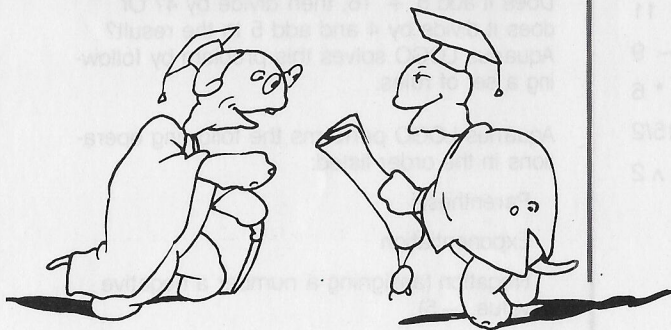
COMMAND	DESIRED ACTION
HATCH "turtle name"	Creates new Turtles
ASK :turtle name	Identifies which Turtle will respond to your new commands
ASK ?	Tells you which Turtle is accepting your commands
DELAY	Creates a brief pause before executing the next command
SHAPE number	Changes the Turtle shape to one selected from the Character Set
TILE number	Changes the Turtle Tracks to one selected from the Character Set
SHAPE 255	Returns the original Turtle Shape, an arrow
TILE 255	Returns the original Turtle Tracks, a line

## CHAPTER 9

# BEYOND TURTLE GRAPHICS NUMBER OPERATIONS

Congratulations! You have now mastered Turtle Graphics in Aquarius LOGO. We hope you've discovered that LOGO is an excellent way to learn programming and is useful in developing your problem solving skills.

In Chapters 9, 10, and 11 you'll be introduced to two fundamental concepts which makes LOGO a total computer programming language. These are the non-graphics features which are numeric calculations, and a feature called List Processing. List Processing is unique to LOGO. It makes programming words and sentences an easier task. By learning to use the features of List Processing, you'll be able to create and sort lists with ease, and to manipulate text and numbers. First, we'll concentrate on the arithmetic functions, then we'll introduce List Processing.



## NUMERIC OPERATIONS IN AQUARIUS LOGO

Most computers have a set of arithmetic commands which allows you to turn your computer into a calculator. Aquarius LOGO has numeric functions so you can add, subtract, multiply, divide, use random numbers, find square roots and calculate exponents.

### MEET THE SYMBOLS FOR ARITHMETIC OPERATIONS

The keyboard uses the following symbols for the math operations in LOGO.

ADDITION	+	$4 + 11$
SUBTRACTION	-	$10 - 9$
MULTIPLICATION	*	$5 * 6$
DIVISION	/	$15/2$
EXPONENTIATION	^	$2 \wedge 2$

Note that a star or asterisk (\*) is used for the multiplication sign and a slash (/) is used for division. For working in decimal numbers, a number is needed on both sides of the decimal point. For example, one-half should be typed as 0.5.

A unique feature of LOGO is performing math operations with your Turtle Graphics commands. For instance, type `FD 5 + 10` and the Turtle will move forward 15 units. Or `RT 360/4` will turn the Turtle 90 degrees.

### ARITHMETIC OPERATIONS — WHICH COMES FIRST?

When you have an expression with several arithmetic operations,  $(5 + 16/4)$ , which operation does the computer perform first? Does it add  $5 + 16$ , then divide by 4? Or does it divide by 4 and add 5 to the result? Aquarius LOGO solves this problem by following a set of rules.

Aquarius LOGO performs the following operations in the order listed:

Parenthesis

Exponentiation

Negation (assigning a number a negative value, -5)

Multiplication and division

Addition and subtraction

Example 1:

$$5 * (6 + 4) / 25 = ?$$

Aquarius LOGO first adds  $6 + 4$  because this operation is in parenthesis. Then it multiplies the result  $10$  by  $5$ . Finally, it divides the result  $50$  by  $25$ . The answer is  $2$ .

Example 2:

$$5 * 6 + 4 / 2 = ?$$

Aquarius LOGO first multiplies  $5 * 6$  with the result  $30$ . Then it divides  $4$  by  $2$  which leaves  $2$ . Next it adds the results of both operations  $30 + 2$  totaling  $32$ .

Example 3:

$$(5 * (6 + 4) / 2) \wedge 2 = ?$$

The first operation Aquarius LOGO performs is adding  $6$  and  $4$ . The expression reads  $(5 * 10 / 2) \wedge 2$ . Next LOGO examines the expression and performs the multiplication and division from left to right. The expression reads  $(50 / 2)$  or  $25 \wedge 2$ . Finally,  $25$  is raised to the second power resulting in  $625$ .

The order in which Aquarius LOGO performs arithmetic operations is called OPERATOR PRECEDENCE.

**NOTE:** When numbers are used in LOGO, they cannot contain commas or special characters like the dollar sign.

\$1,250.75 ..... NO

1250.75 ..... YES

## WORKING WITH INTEGER AND RANDOM

The Integer or INT (number) functions give you only the integer for an answer when there are fractions of the number. For example:

INT 23.625

23

INT 3.124

3

The random or RN (number) function chooses a number between zero ( $0$ ) and one less than the number you select. For example, if you choose  $15$ , the random number could be any number from zero to  $14$ .

RN 15

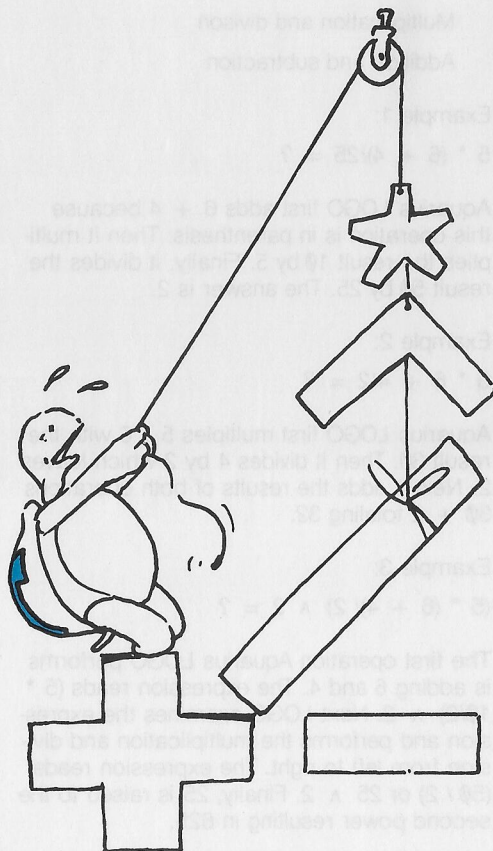
4

Random is useful when you want the computer to select a number to use in a command. This is why INK (RN 16) will randomly select a Turtle color number between INK 0 and INK 15. In the Random Walk procedure, we asked Aquarius to select a heading from a range of random numbers between zero and 361 degrees (HD RN 361).

RN 361

41

We briefly introduced the most commonly used math operations in LOGO.



## CHAPTER 10

# BEYOND TURTLE GRAPHICS: LIST PROCESSING

List Processing is a structure for organizing letters, words, and numbers just as a catalog system organizes the books in a library. You can store your own information in a list. Then you can add, retrieve, and edit this information with ease.

### WHAT IS LIST PROCESSING?

List Processing is a means of organizing and using 'text'. The basic organizational structure for organizing 'text information' is the familiar square bracket.

**[ text information ]**

You recall that we used square brackets with the following LOGO commands:

```
POS [ 40 30 ]
REP 4 [ FD 10 RT 90 ]
IFTRUE [ PRINT :X ] or
IFFALSE [ FD 10 LT 45 ]
```

What do we mean by 'text'? "Text information" in LOGO is found in four different forms: numbers, character strings, words, and LISTS. Using numbers in LOGO was discussed in Chapter 9. Let's examine the remaining three types of 'text'.

## CHARACTER STRINGS

A string is a series of characters which includes letters, punctuation marks, and spacing. Character strings have no numeric value and are treated 'literally' by LOGO. Strings are always enclosed by two apostrophe marks ( ' '). An example of how strings are used is the TP command. Make sure you are in the graphics mode or split screen. You will use a CHARACTER STRING when you want the computer to print exactly what you typed in. Type:

```
TP 'HELLO, HOW ARE YOU?'
```

The Turtle printed your greeting on the screen, literally!

## WORDS

A word is formed by a series of characters and/or numbers. However, spaces are not permitted within a word. Each word is a unit of information. Words are always introduced by " (quotation mark). For example, when we assign a value to a variable name or word, we use the MAKE command followed by a quote.

```
MAKE "HELLO "HAPPY
```

The value for the word HELLO is now the word HAPPY. Type:

```
:HELLO
```

and the computer returns:

```
HAPPY
```

## LISTS

A list is a sequence of words in LOGO. Each word is separated by a space. Lists are always preceded by a square bracket and end with a square bracket. In this format, LOGO is unique because it enables you to work with 'text' on a word by word basis.

A list in LOGO is organized according to the position of the character or word. It is similar to waiting in line. The first person in line is in the first position, the second person is in the second position, etc. LOGO looks at the information whether it be character strings or words and organizes it according to its position in a LIST. LIST is 'text' organized inside a square bracket. You will use LISTS when you want to organize words in a particular order.

As an example, let's create a LIST of words called NOUNS. Type:

```
MAKE "NOUNS [ CATS DOGS BIRDS ]
```

If you type the variable name :NOUNS, the following LIST will appear. Type:

```
:NOUNS  
[ CATS DOGS BIRDS ]
```

To select the first word in the LIST, LOGO has a command called **FIRST**. Type:

```
FIRST :NOUNS  
[ CATS ]
```

Notice LOGO returned with the first word in the LIST. Type the next command, **BF** for But First. Type:

```
BF :NOUNS  
[ DOGS BIRDS ]
```

A LIST was returned but the first word was not there. The BF or But First command returns a list without the first word.

Try the next command, **LAST**. Type:

```
LAST :NOUNS  
[ BIRDS ]
```

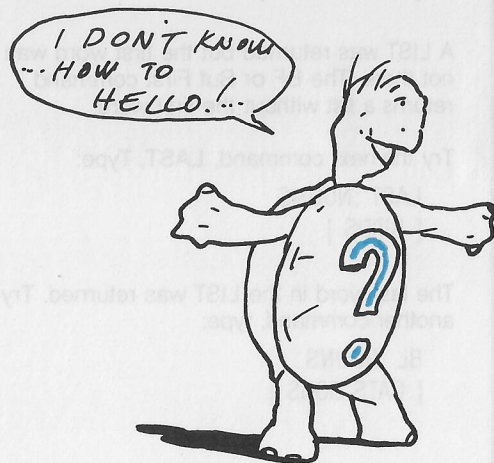
The last word in the LIST was returned. Try another command, type:

```
BL :NOUNS  
[ CATS DOGS ]
```

By now you've guessed the BL stands for But Last which returns the entire LIST without the last word. You've probably also noticed that LOGO is very fussy about how punctuation marks are used.

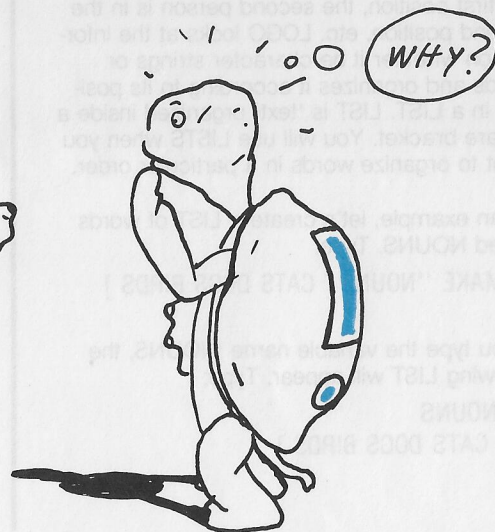
To help the computer distinguish whether you are using character strings, words, or lists, certain punctuation marks are used. To illustrate, try this exercise. Type:

HELLO



LOGO responds, "I DON'T KNOW HOW TO HELLO". Why?

LOGO is confused because it doesn't know whether HELLO is a list, a string, a word, a previously defined procedure, a name for a variable, or a variable itself. With the aide of punctuation marks, Aquarius LOGO can accurately recognize what HELLO is, and follow the instructions for HELLO.



Let's study the punctuation marks.

## NO PUNCTUATION

If HELLO is typed without any punctuation marks, you are telling Aquarius LOGO to carry out a procedure named HELLO. If HELLO was previously defined, the procedure would be followed.

## ' message ' apostrophes

When apostrophes are used with the TP command, the computer prints exactly the message enclosed by the apostrophes. We call this a character string. For example, TP '4 + 4' will print 4 + 4 on the screen.

## “ quotation mark

The quotation mark identifies a word. When “ is used with the MAKE command, it gives a value to a variable (a word). For example, in MAKE “HELLO 20, HELLO is assigned the value of 20. To print a word on the screen by using the PRINT command, a quotation mark must be used followed by a word. For example, PRINT “HELLO.

## : dots

: (dots) instruct Aquarius LOGO to generate the latest value assigned to a variable. If we type :HELLO, the value 20 will appear on the screen. The value for a variable is not limited to numbers only. Strings, words, and LISTS can also be used. For example, if you reassign another value such as MAKE “HELLO [ 20 30 ], the value of HELLO is updated to [ 20 30 ]. Typing :HELLO RTN will give you [ 20 30 ].

---

( ) parentheses

Parentheses organize a group of words, strings, or numbers that work together. An example is  $2*(5 + 10)$

[ ] brackets

Brackets are used to enclose messages, identified as a list. Recall that the REP, POS, IFTRUE and IFFALSE commands all use brackets. These four commands only function when LISTS are used. For example:

```
?PRINT [ TO BE OR NOT TO BE ]  
[ TO BE OR NOT TO BE ]
```

To remove the brackets from your print command, use the PRINTSE command. Type:

```
?PRINTSE [ TO BE OR NOT TO BE ]  
TO BE OR NOT TO BE ?
```

The question mark indicates that LOGO is ready to accept the next text line. To move the question mark to the beginning of the new line, use the PRINTNL command. Type:

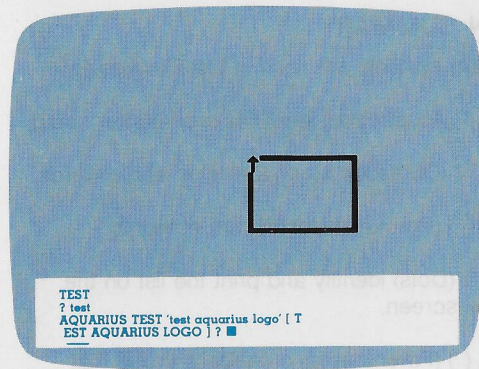
```
?PRINTNL [ TO BE OR NOT TO BE ]  
[ TO BE OR NOT TO BE ]  
?
```

To illustrate WORD, STRING, and LIST with their respective punctuation symbol, use the same term 'TEST', by typing the following short program.

```
TO TEST
REP 4 [ RT 90 FD 15 ]
MAKE "TEST "AQUARIUS
PRINT :TEST
PRINT "TEST
PRINT 'test aquarius logo'
PRINT [ TEST AQUARIUS LOGO ]
END
```

Notice that "TEST" is used differently with each set of punctuation marks. Typing TEST without any punctuation marks will run the procedure TEST. Type TEST:

The first instruction draws a box on the Graphics Screen and displays a series of messages in the Command Window.



The next two lines assign the value Aquarius to the word TEST. The word TEST is printed because a quotation mark was used with the PRINT command. 'Test aquarius logo' is a STRING because an apostrophe was used with the PRINT command. Note that the message within the apostrophe was not capitalized, (unless capitals are used). LOGO identifies STRINGS as they appear. The last line uses square brackets, meaning that [ TEST AQUARIUS LOGO ] is a LIST with words that can be evaluated word by word.

## IN SUMMARY

Five symbols are used in List Processing:

1. ' (Apostrophe) identifies a character string.
2. " (Quotation Mark) identifies single words.
3. [ ] (Brackets) identify lists of words.
4. : (Dots) identify and print the list on the textscreen.
5. ( ) (Parentheses) organize a group that works together.

These few commands and punctuation symbols illustrate what LOGO can do with LIST PROCESSING. List Processing organizes and edits words or characters, primarily according to their position in a LIST. The rest of Chapter 10 will introduce you to the other commands in LOGO which show you how to create LISTS, how to select words from a LIST, how to add words to LISTS, and how to write programs than enable dialog with the computer. Chapter 11 will show you how to put it all together by presenting model programs that use many of the List Processing commands you'll learn in this chapter.

## LIST PROCESSING AT WORK

Here's a short program which demonstrates one way to edit text using the BF (but first) command. In addition, conditional branching IFTRUE is introduced to end a loop. Type:

```
TO SHORTEN :ALIST
:ALIST
IFTRUE [ PRINTSE :ALIST NL
SHORTEN BF :ALIST ]
END
```

To run the program, type:

```
FS
SHORTEN [ TO BE OR NOT TO BE ]
```

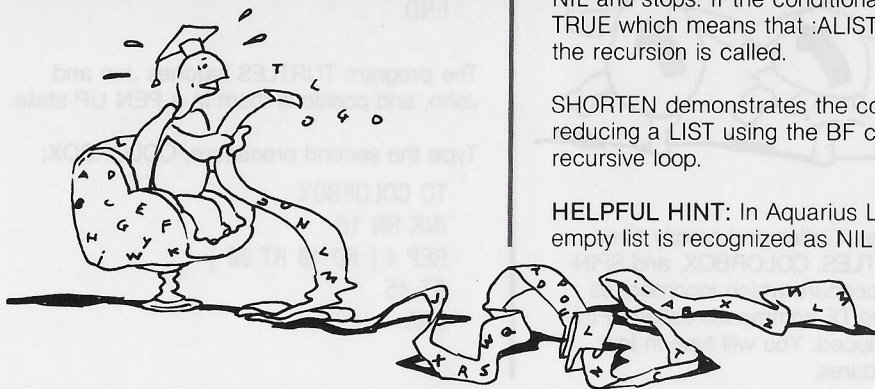
FS (Full Screen) gives you the entire screen as the text window. Type any text you want in brackets.

Now you see:

```
TO BE OR NOT TO BE
BE OR NOT TO BE
OR NOT TO BE
NOT TO BE
TO BE
BE
NIL
?
```

To return to a split screen, use the SS command.

SHORTEN is a program which can be applied to many other situations. The beauty of the



program lies in its use of the BF (but first) command to remove words from a LIST. Words are removed according to their position in the LIST. BF removes the first word from the list and prints the remainder of the LIST. The recursive call, SHORTEN BF is the critical instruction because it repeats the program and the LIST each time. As the program proceeds through the LIST a new word falls into the first position and is then removed. In this program, the seventh time the program is repeated, the LIST becomes empty. The term 'empty brackets' or 'empty LIST' is used to describe this condition.

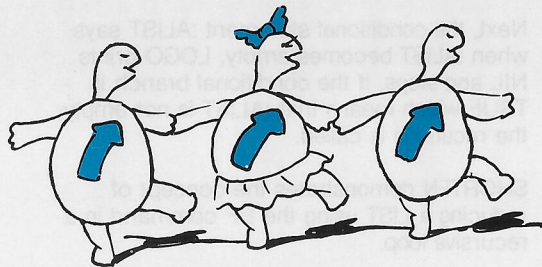
Next, the conditional statement :ALIST says when :ALIST becomes empty, LOGO prints NIL and stops. If the conditional branch is TRUE which means that :ALIST is not empty, the recursion is called.

SHORTEN demonstrates the concept of reducing a LIST using the BF command in a recursive loop.

**HELPFUL HINT:** In Aquarius LOGO, an empty list is recognized as NIL and vice versa.

## SPINBOXES

The joy in programming with Aquarius LOGO is being able to use one idea in a text mode, then in another example, apply the same idea in a graphics mode. In our next example, we will demonstrate how the BF command in a recursive loop, can direct many Turtles in drawing a familiar program, SNOWFLAKES.



We will use three Turtles and modify three programs: TURTLES, COLORBOX, and SPINBOX. A new procedure which incorporates the BF, ASK and TF commands called SPINBOXES is introduced. You will type in four separate procedures.

Before we proceed, you may need to RESET the system to empty the workspace for more memory.

Type the first procedure, TURTLES:

```
TO TURTLES
PEN Ø
HATCH "JAN
HATCH "JOHN
ASK :STUDS
POS [ 4Ø 15 ]
ASK :JAN
POS [ 6Ø 45 ]
ASK :JOHN
POS [ 2Ø 45 ]
END
```

The program TURTLES hatches Jan and John, and positions them in a PEN UP state.

Type the second procedure, COLORBOX:

```
TO COLORBOX
INK RN 16
REP 4 [ FD 1Ø RT 9Ø ]
RT 45
END
```

COLORBOX instructs the Turtles to draw a colorful box.

Instead of instructing each Turtle separately, the next procedure SPINBOX will instruct all of them in one procedure. COLORBOX uses the BF format similar to the SHORTEN program. In addition, a new command TF (Turtle Family) is introduced.

Type the third procedure, SPINBOX:

```
TO SPINBOX :ALLT
:ALLT
IFTRUE [ ASK FIRST :ALLT PEN 1
COLORBOX SPINBOX BF :ALLT ]
END
```

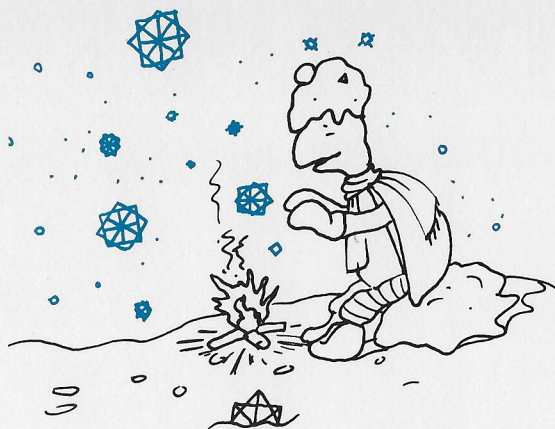
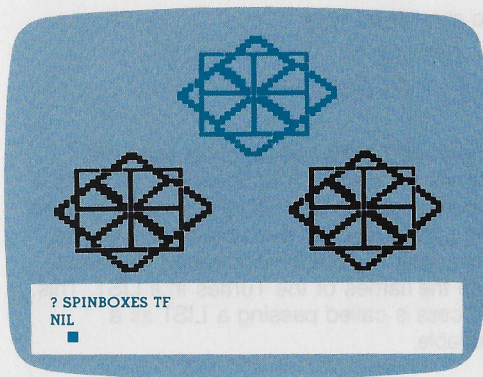
You'll recall in the SHORTEN program, the variable :ALIST enabled you to type in any LIST of words in brackets. In SPINBOX a variable list called ALL TURTLES or ALLT is used instead. Since ALLT requires a list, you will use the TF command (Turtle Family) to create a list.

To see the list type:

```
TURTLES TF
[ STUDS JAN JOHN ]
```

The three Turtles are hatched and their names are placed in an exclusive LIST. As a result, when the SPINBOX command is used, the LIST, :ALLT will be substituted with the command TF. The TF command will substitute the names of the Turtles in a LIST. This process is called passing a LIST as a variable.





## ADD TO SPINBOX

To create an endless series of snowflakes, add this procedure. Type:

```
TO SPINBOXES :ALLT
  SPINBOX :ALLT
  SPINBOXES :ALLT
END
```

SPINBOXES reassigns the command TF as the variable to :ALLT. To run the program, type:

```
SPINBOXES TF
```

Time and programming space is saved by using a procedure such as SPINBOX to speak to each Turtle. Otherwise, an ASK command must be used each time you instruct a Turtle.

## COMMANDS TO ADD STRINGS AND WORDS TO CURRENT LISTS

There are times when you want the option of adding words to an existing list. To do this, first create a list. Type:

```
MAKE "nouns [ SUN MOON STARS PLANETS ]
```

Now you can use the FPUT or LPUT command. Place the new word or words in [ ] after the FPUT or LPUT command. In our example, we are adding two words, meteor and comet, to the list :nouns.

```
FPUT [ meteor comet ] :nouns  
[ [ meteor comet ] sun moon stars planets ]
```

FPUT adds two words to the beginning of an existing list of words.

```
LPUT [ meteor comet ] :nouns  
[ sun moon stars planets [ meteor comet ] ]
```

LPUT adds two words to the end of an existing list of words.

Words can also be added using a quotation mark. Type:

```
FPUT "TASTY [ 10 APPLES AND 20 ORANGES ]  
[ TASTY 10 APPLES AND 20 ORANGES ]
```

To edit a LIST by substituting one word for another, two commands can be used in combination, FPUT and BF. Type:

```
FPUT 5 BF [ 10 APPLES AND 20 ORANGES ]  
[ 5 APPLES AND 20 ORANGES ]
```

BF removes 10 from the LIST and with FPUT substitutes the number 5. Type:

```
FPUT 15 FPUT "MELONS BF BF [ 10 APPLES ]  
[ 15 MELONS ]
```

The two BF commands removed 10 APPLES from the LIST, and the two FPUT commands substituted 15 MELONS.

## INTERACT WITH THE COMPUTER

To write an interactive program where the computer prompts you with a question, then waits for an answer, the command RL or "readlist" is used. The RL command tells Aquarius to wait for the user to type an answer.

To see how RL works, type:

RL

Aquarius LOGO waits for a response from the keyboard. Notice there isn't a ? (question mark) before the cursor. Type:

a 10 b

The computer puts your input into a LIST.

[ a 10 b ]  
?

Another command that works similarly to RL is RC or Read Character. RC also places the computer in a pause mode, waiting for the

user to type in a response. RC however, returns a character string after a letter is typed. Type:

RC

Aquarius pauses, waiting for you to type a response. If you press the letter A, LOGO will respond with:

'A'

'A' is a character string and is enclosed by two apostrophes. The usefulness of the RC will become evident when we introduce you to a game program called GUESS A LETTER in Chapter 11.

Study the following procedure which prompts the user to place numbers in the program. Once the numbers are placed in the program, a figure is drawn. Type:

TO REQUEST

PRINTSE [ WHAT IS THE LENGTH? ]

MAKE "LENGTH FIRST RL

PRINTSE [ HOW MANY TURNS? ]

MAKE "TURNS FIRST RL

REP :TURNS [ FD :LENGTH RT 360/:TURNS ]

END

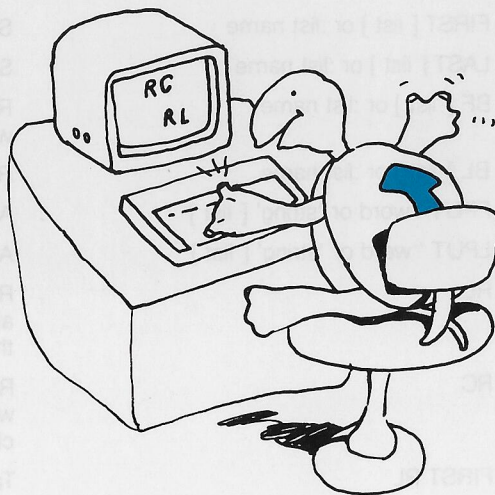
To run this program, type:

## REQUEST

The procedure prompts you with the question "WHAT IS THE LENGTH?" The answer typed will become the value for the variable :LENGTH. The next question asks, "HOW MANY TURNS?" The answer becomes the value for the variable :TURNS. Once your answers are given (be sure to press **RTN** after each number), LOGO draws a figure. If you type REQUEST, and respond with 10 for length and 8 for turns, an octagon will be drawn.

How does this all take place? The line MAKE "LENGTH FIRST RL consists of three actions. One, the variable LENGTH is created. Next the RL function is used with the FIRST command. The number you type in will become a LIST. The RL command not only waits for you to give an answer, it also places that answer in a LIST. For example, typing in 10 will result in [ 10 ]. The FIRST command will then pull out the number 10 from the LIST, and assign its value to the variable LENGTH. The same process occurs with TURNS.

In review, both the RL and RC commands enhance your ability to write interactive programs with Aquarius LOGO.



---

## SUMMARY

Creating and selecting words, strings, or sentences are based on the word's position in a LIST. The following commands were discussed in Chapter 10.

FIRST [ list ] or :list name

Selects the first word or string from a LIST

LAST [ list ] or :list name

Selects the last word or string from a LIST

BF [ list ] or :list name

Returns all the words in a LIST except the first word

BL [ list ] or :list name

Returns all the words in a LIST except the last word

FPUT "word or 'string' [ list ]

Adds the word or string to the beginning of a LIST

LPUT "word or 'string' [ list ]

Adds the word or string to the end of a LIST

RL

Reads a line of keyboard input terminated by RTN and returns a LIST. RL waits for a response from the user.

RC

Reads a single character from the keyboard. RC waits for a response from the user and returns the character as a STRING

FIRST RL

Takes the first answer entered from the LIST created by RL.

PRINTSE

Prints a LIST or STRING without punctuation marks.

PRINTNL

Prints a LIST or STRING and then moves the cursor to the next line.

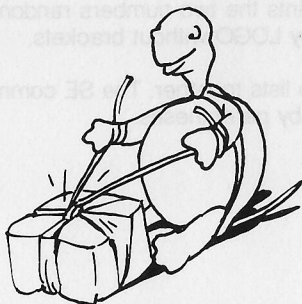
NL

Moves the cursor to the beginning of the next line.

---

## CHAPTER 11

# PUTTING IT ALL TOGETHER



Now that you've been introduced to the List Processing commands, you can create, print, and select numbers, character strings, words, or sentences from lists. To understand how List Processing works, we are going to present three programs for 'putting it all together'. The first program is called MULTIPLICATION DRILL, the second GUESS A LETTER, and the third GUESS A WORD.

### MULTIPLICATION DRILL

Creating a math drill for any operation consists of presenting a series of problems with random numbers, accepting an answer and checking to see if the answer is correct.

There are four parts to MULTIPLICATION DRILL.

1. Present a multiplication problem, using two randomly selected numbers between zero (0) and 9.
2. Ask the user to give the answer. LOGO will check the answer.

3. If the answer is correct, LOGO will give a message, "THAT'S RIGHT!" If the incorrect answer is given, LOGO will give the correct answer with the message, "THE ANSWER IS ---."

4. Present the next problem.

You will use two procedures to create the Multiplication Drill.

1. The first creates the variables, prints the multiplication problem, and accepts an answer from the player. This procedure is called DRILL.

Type:

```
TO DRILL
MAKE "NA RN 10
MAKE "NB RN 10
PRINTSE (SE :NA [ * ] :NB [ = ? ] )
MAKE "ANSWER FIRST RL
MAKE "CORRECT :NA * :NB
COMPARE
DRILL
END
```

In the DRILL program, the following variables and Logo commands are used:

- NA**            Number 'A', the first random number selected.
- NB**            Number 'B', the second random number selected.
- ANSWER**       The answer to the problem given by the player.
- CORRECT**      The answer calculated by LOGO.

RN asks LOGO to select a random number from zero to nine.

FIRST RL receives an answer from the keyboard, and assigns the value to :ANSWER.

PRINTSE prints the two numbers randomly generated by LOGO without brackets.

SE joins two lists together. The SE command is enclosed by parentheses.

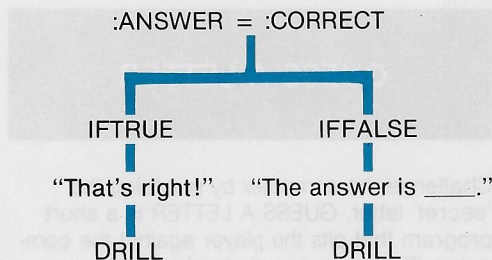
2. The second procedure takes the answer provided, compares the player's answer with the actual answer to the problem, and prints a message. The procedure will be called COMPARE.

Add the COMPARE procedure. Type:

```
TO COMPARE
:ANSWER = :CORRECT
IFTRUE [ PRINTSE 'That's right!'NL ]
IFFALSE [ PRINTSE 'The answer is 'PRINT
:CORRECT NL ]
END
```

**HELPFUL HINT:** You can type up to 45 characters of text without using the **RTN** key. This is equal to one and one-quarter lines of material. If the cursor stays in one spot, you have reached the end of the line. Just press the **RTN** key and you can continue writing your program. Don't press the **RTN** key in the middle of a word or LOGO will think there are two words instead of one.

In the COMPARE procedure, the test condition `:ANSWER = :CORRECT` is established. If the test is true, the player is rewarded with a "THAT'S RIGHT!" If the test is false, the correct answer is given. Notice that NL was used to advance the cursor to the next line.



Type DRILL to start the program. Each time the question is displayed, just type in your answer and press the **RTN** key. For example:

DRILL

$5 * 6 = 30$

That's right!

$4 * 5 = 10$

The answer is 20

To end the program, you may use the WARM START, or press the **RTN** key before entering your answer.

## GUESS A LETTER

Challenge the computer by guessing the 'secret' letter. GUESS A LETTER is a short program that pits the player against the computer. The computer randomly selects a mystery letter. The player must guess the mystery letter. After you have guessed a letter LOGO gives you a clue by displaying 'Too low or Too high'. When the correct answer is given, LOGO rewards you by saying, 'That's Great!'

Two procedures are needed for GUESS A LETTER. The first one, CREATE generates a random secret letter from A to Z. The second procedure named COMPARE checks the mystery letter with the player's guess. A new command CH for Character is used. CH converts a number code between 0 to 255 to a specific character, and also performs the reverse. For example, CH 97 is character 'a', and CH 'a' will be number 97.

**NOTE:** Due to the complexity of this and the remaining programs, you may need to have additional memory to accommodate the entire program. However, you can type in one

procedure at a time to examine the programming strategies involved.

Four variables are created.

1. NUM stands for a random letter number from the character set.
2. GUESS is the variable for 'letter to be guessed'.
3. ANSWER is a variable to accept the player's response.
4. DIFF is the variable for the difference between the two compared letters.

Type:

```
TO CREATE
  PRINTSE 'GUESS A LETTER'
  NL
  MAKE "NUM ( 97 + RN 26 )
  MAKE "GUESS CH :NUM
  COMPARE
  END

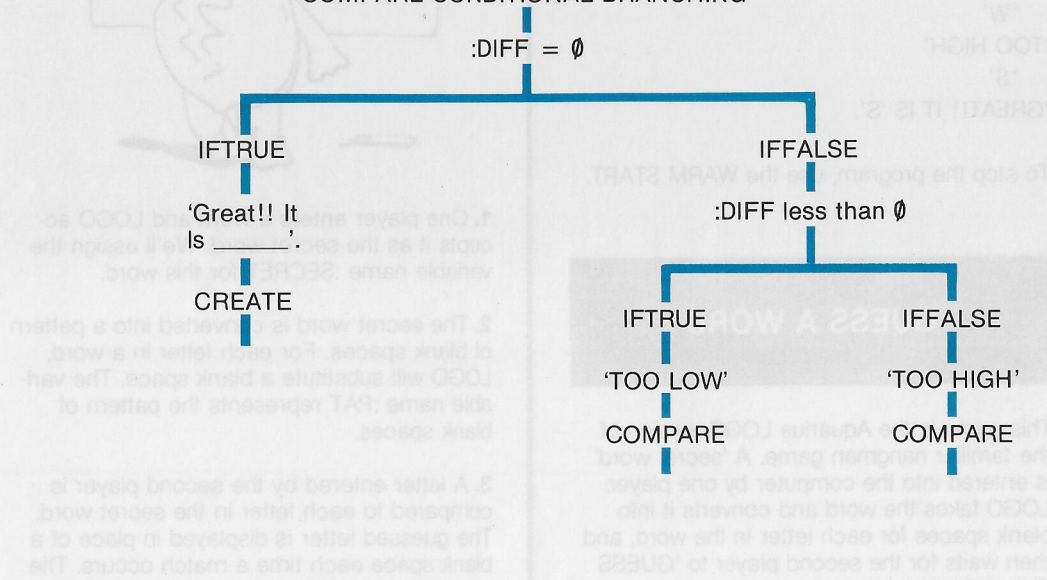
TO COMPARE
  MAKE "ANSWER RC
  PRINTNL :ANSWER
  MAKE "DIFF ((CH :ANSWER)-(CH :GUESS))
  :DIFF = 0
```

```

IFTRUE [ PRINTSE 'GREAT!! IT IS ' PRINT :ANSWER NL CREATE ]
IFFALSE [ :DIFF < 0 IFTURE [ PRINTSE 'TOO LOW' NL ]
IFFALSE [ PRINTSE 'TOO HIGH' NL ]
COMPARE ]
END

```

### COMPARE CONDITIONAL BRANCHING



**NOTE:** The alphabet is organized with the letter "A" on the low end, and 'z' on the high end.

Type CREATE to run the program. As an example:

CREATE

Guess a letter

'A'

'TOO LOW'

'W'

'TOO HIGH'

'S'

'GREAT!! IT IS 'S'.

To stop the program, use the WARM START.

## GUESS A WORD

This game is the Aquarius LOGO version of the familiar hangman game. A 'secret word' is entered into the computer by one player. LOGO takes the word and converts it into blank spaces for each letter in the word, and then waits for the second player to 'GUESS THE WORD'. Writing the program involves four steps.



1. One player enters a word and LOGO accepts it as the secret word. We'll assign the variable name :SECRET for this word.
2. The secret word is converted into a pattern of blank spaces. For each letter in a word, LOGO will substitute a blank space. The variable name :PAT represents the pattern of blank spaces.
3. A letter entered by the second player is compared to each letter in the secret word. The guessed letter is displayed in place of a blank space each time a match occurs. The variable name :TRY represents the guessed letter.

4. When all the guessed letters match the secret word, the message 'WINNER' is displayed. If the letters don't match, the player is asked to "ENTER A LETTER" and continue guessing.

We are now ready to write the LOGO procedures for the game. Type:

```
TO START
  PRINTSE [ ENTER A WORD ]
  MAKE "SECRET RL
  REP 24 [ NL ]
  MAKE "PAT CONVERT :SECRET
  PRINTNL :PAT
  GUESS
END

TO CONVERT :ALIST
:ALIST
IFTRUE [ FPUT " CONVERT BF
:ALIST ]
END
```

These two procedures accomplish the first two steps in designing GUESS A WORD. The START procedure prompts the player to ENTER A WORD, and accepts the secret word as input. The procedure CONVERT takes the secret word and converts it to the correct number of blank spaces. The underscore key, (Shift dash) is used for the blank space.

Notice that we are passing the secret word :SECRET to the variable :ALIST in the CONVERT procedure. This you'll recall is 'value passing'.

The RL command in the instruction MAKE "SECRET RL places the secret word as a variable into a LIST. The cursor is advanced 24 times to hide the secret word from the second player. CONVERT changes the secret word into a pattern of blank spaces. This principle is similar to the discussion in Chapter 10 on letter and word substitutions.

To check your procedure, type START. After the message to ENTER A WORD, type in any word, however be sure to LEAVE A SPACE between each letter entered. This enables LOGO to examine each letter as a separate unit in the LIST, rather than the whole word

at once. After the secret word is typed in with spaces, press the `[RTN]` key. The program will look like this:

```
START
ENTER A WORD T O P
```

```
[ _ _ _ ]
```

A message, "I DON'T KNOW HOW TO GUESS NEAR START" will appear. Don't be concerned. We will teach LOGO how to GUESS in the next procedure.

Type:

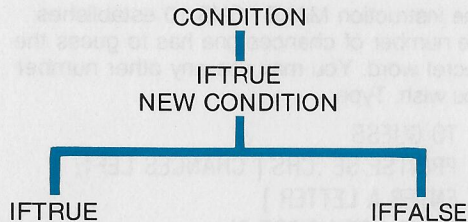
```
TO GUESS
  PRINTSE [ ENTER A LETTER ]
  MAKE "TRY FIRST RL
  MAKE "PAT COMPARE :SECRET :PAT :TRY
  NL
  PRINTNL :PAT
  :PAT = :SECRET
  IFTRUE [ PRINTSE [ WINNER! ] NL ]
  IFFALSE [ GUESS ]
END
```

To complete the project, type in the COMPARE procedure.

```
TO COMPARE :SECRET :PAT :TRY
:SECRET
IFTRUE [ :TRY = FIRST :SECRET
IFTRUE [ FPUT :TRY COMPARE
BF :SECRET BF :PAT :TRY ]
IFFALSE [ FPUT FIRST :PAT COMPARE
BF :SECRET BF :PAT :TRY ] ]
END
```

GUESS prompts the second player to enter a letter to guess the secret word. The instruction FIRST RL takes the letter and assigns it to the variable :TRY. The next line compares the letter with the secret word, and fills in the blank space if the match is made. Now the computer goes to the Compare procedure.

On the surface, this may appear to be a rather complex procedure. The concept of double conditional branching is used. Double branching takes one condition, and one outcome, IFTRUE, and sets up another set of branches or options.



The first condition checks whether the first letter in the secret word :SECRET is identical to the first letter of the guessed word :TRY = FIRST :SECRET.

Next the letters are matched and if the condition is true, the guessed letter :TRY is placed in its correct position. This is done with the FPUT command. If the letters don't match, then a blank space or a previously correct guess is returned using FPUT FIRST :PAT. In the recursion statement, COMPARE, the LIST for :SECRET and :PAT are reduced at the same time. This enables LOGO to check each guessed letter with each letter in the secret word.

The recursion ends when the LIST containing the secret word :SECRET is empty. The correct letters or blank spaces are simultaneously returned, i.e. FPUTed into the pattern :PAT.

When the computer completes the Compare procedure, the program returns to the Guess procedure. When the secret word is matched, WINNER is printed. If not, the GUESS procedure is repeated.

The GUESS A WORD program is now complete. To play the game, type START and press RTN. Be sure to type a space between the letters when entering a secret word. Here's what the game should look like.

```
START
ENTER A WORD  L O G O
```

```
[ _ _ _ _ ]
ENTER A LETTER  O
```

```
[ _ 0 _ 0 ]
ENTER A LETTER  T
```

```
[ _ 0 _ 0 ]
ENTER A LETTER  L
```

```
[ L 0 _ 0 ]
ENTER A LETTER  G
```

```
[ L O G O ]
WINNER!
```

## GUESS A WORD WITH LIMITED CHANCES

To increase the difficulty level of the game, one can limit the number of times a player can guess the secret word. First we establish a new variable to accept the number of chances a person has to guess the word. The variable name is :CHS for chances. The chance instruction uses a conditional branch. If the guessed letter and the :PAT are the same, a letter is placed in the correct position. If a guess is missed, the remaining chances are reduced by one.

To include this feature, we will modify the START and GUESS program. A new procedure LOSE is introduced. CONVERT and COMPARE will remain the same.

```
TO START
NL PRINTSE [ ENTER A WORD ]
MAKE "SECRET RL
REP 24 [ NL ]
MAKE "PAT CONVERT :SECRET
PRINTNL :PAT
MAKE "CHS 10
GUESS
START
END
```

The instruction MAKE "CHS 10 establishes the number of chances one has to guess the secret word. You may use any other number you wish. Type:

```
TO GUESS
PRINTSE SE :CHS [ CHANCES LEFT,
ENTER A LETTER ]
MAKE "TRY FIRST RL
MAKE "PAT COMPARE :SECRET :PAT :TRY
NL
PRINTNL :PAT
:PAT = :SECRET
IFTRUE [ PRINTSE [ 'WINNER! ] NL ]
IFFALSE [ :PAT = :PAT IFTRUE
[ MAKE "CHS :CHS-1 ]
:CHS = 0
IFTRUE [ LOSE ]
IFFALSE [ GUESS ] ]
END
```

```
TO LOSE
PRINTSE [ YOU LOSE, THE WORD IS ]
NL
PRINTNL :SECRET
NL
END
```

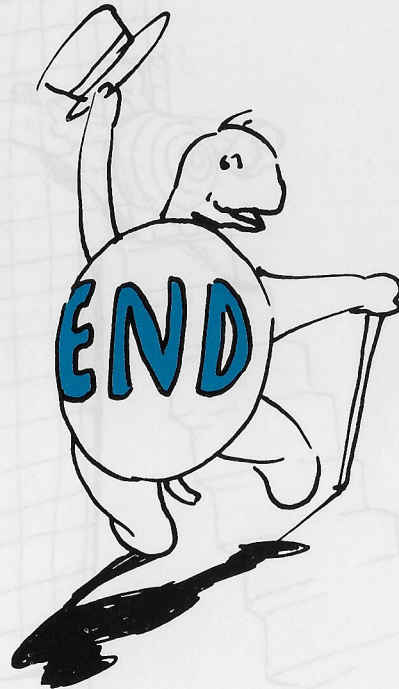
---

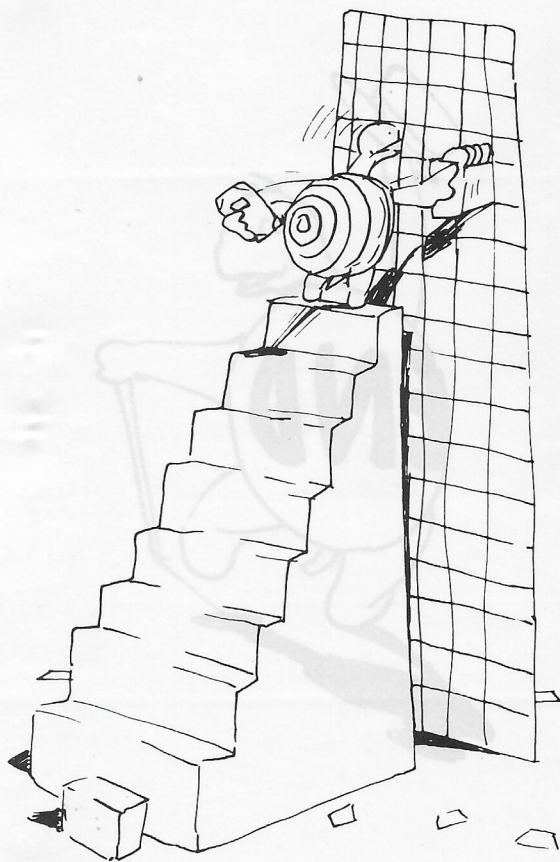
An additional modification to GUESS is to subtract the value of CHS by 1 when the guessed letter is incorrect. This is found in the instruction MAKE "CHS :CHS-1. A new prompt, YOU LOSE is printed when CHS reaches zero.

## IN CLOSING

In this chapter, many concepts were introduced to you that may be new. Take your time with each section, and review it at your leisure. Remember to start with the larger problem, break it into its parts, and test each portion. LOGO is an extremely flexible language. By creating your own programs, LOGO will grow with you.

Enjoy learning with Aquarius LOGO!





## LOGO REFERENCE SECTION

## TABLE OF CONTENTS

### APPENDIX A:

#### TURTLE TIPS

#### COMMAND INPUTS

#### SCREEN COORDINATES

#### STARTING, STOPPING, AND RESTARTING LOGO

#### REPRESENTATION OF NUMBERS

#### OPERATORS

- Numeric Operators
- Relational Operators
- Operator Precedence

#### MEMORY REQUIREMENTS

- Workspace
- Edit Buffer
- Words and Strings
- Lists
- Stacks

Programming Tips for Using Recursion  
Coping with NO ROOM

### TURTLE PROGRAMMING TIPS

- Plotting Turtle Tracks
- Erasing Turtle Tracks
- What to Do When the Turtle Won't  
Budge
- Turtle Shadows

### APPENDIX B:

#### LOGO COMMAND SUMMARY

### APPENDIX C:

#### RESERVED WORDS

### APPENDIX D:

#### AQUARIUS LOGO CHARACTER SET

### APPENDIX E:

#### LOGO ERROR MESSAGES

#### 90-DAY LIMITED WARRANTY

---

## APPENDIX A

---

### TURTLE TIPS

Logo allows you to talk to one turtle at a time. The turtle that you are currently talking to is called the “current turtle.”

By default, when you start Logo, STUDS is the current turtle. When the arrow first appears in the middle of your screen, STUDS has the following characteristics or “properties.”

#### PROPERTY

Position:  
Heading:  
Turtle state  
Pen state  
Turtle shape  
Tile shape  
Ink color

#### DEFAULT

[ 40 30 ]  
90  
TURTLE 1 (turtle displayed)  
PEN 1 (pen down)  
SHAPE 255 (arrow)  
TILE 255  
INK 0 (black)

As you hatch additional turtles, you use the ASK command to tell Logo which turtle you want to talk to. Each turtle that you hatch assumes the “properties” of the current turtle.

For example, suppose that PEN 0 is in effect when you hatch a turtle. The pen state for the new turtle will also be PEN 0.

## COMMAND INPUTS

Most of the Logo commands require an input — some value or list of values that the command uses to perform its function. For example, the PEN command uses a number or a question mark as its input: PEN 1 sets the pen down, and PEN 0 lifts the pen up. When you use a question mark instead of a number, PEN tells you what the current pen state is.

In this reference section, command names are shown in uppercase, and inputs are shown in lowercase. Type the command name exactly as shown. Then, type the inputs indicated for the command.

For example, the following description of the PEN command means that you enter the command name PEN, one or more spaces, and a pen state number or a question mark.

PEN number

or

PEN ?

Type special characters shown with inputs. For example, type brackets to enclose the list of numbers that represent X,Y coordinates in the POS command:

POS [ column row ]

Some commands can take more than one input. For example, the LIST command returns a list of objects identified in the command. Objects can be numbers, strings, words, or lists. Ellipsis (a string of three periods) means that the item can appear more than once in the command.

LIST object...

---

## SCREEN COORDINATES

The turtle's position on the Graphics Screen is represented by X,Y coordinates. The Graphics Screen has 80 vertical columns and 60 horizontal rows. The X coordinate corresponds to vertical columns, and the Y coordinate corresponds to horizontal rows.

Columns are numbered 0-79 from left to right, and rows are numbered 0-59 from the top of the screen to the bottom. POS [ 0 0 ] is the upper left-hand position on the Graphics Screen, and POS [ 79 59 ] is the bottom right-hand position on the Graphics Screen.

## STARTING, STOPPING, AND RESTARTING LOGO

When you turn your Aquarius computer on, Logo starts "cold" with an empty memory. As you use Logo, you fill Logo's memory with the procedures and variables that you define. This information remains in Logo's memory until you turn the power off.

Turning the power off is a quick way of erasing memory. However, you can do the same thing by pressing **RST** and then **RTN**. Turning the power on — or pressing **RST** and then **RTN** — is called a "cold start."

A "warm start," on the other hand, allows you to interrupt whatever Logo is currently doing without erasing Logo's memory. You warm-start Logo by pressing **RST** and then **CTL-C**.

---

---

You use a warm start, for example, to discard the new version of a procedure that you are currently editing — without erasing any of the other procedures in your workspace. You can also use a warm start to return to the command mode immediately when Logo is slow to respond to **CTL-G** to halt a program.

What do you do when you inadvertently press the **RST** key? Pressing **RST** returns you immediately to Logo's title page. If you press **RTN** next, you could start Logo and erase Logo's memory. To return to the command mode and save Logo's memory, press **CTL-C** instead of **RTN**.

A warm start is also useful when you want to return to the command mode after entering an LP, CHECK, LOAD, or SAVE command.

Suppose, for example, that you wanted to save some Logo procedures on cassette. But, you inadvertently pressed **RTN** before you pressed the **PLAY** and **RECORD** keys on your recorder — or you realized in the middle of saving procedures that the recorder cable was not connected to your Aquarius computer.

When you use the SAVE command, Logo doesn't know whether a cassette recorder is connected to your Aquarius computer. Logo returns control to you after Logo thinks that it has saved the last procedure or variable in the SAVE list. You can allow Logo to continue, or you can warm start Logo to return to the command mode. Then, you can reenter the SAVE command.

The LP and LOAD commands work a little differently.

When you enter the LP command to print procedures on your Aquarius printer, Logo asks the printer if it is ready to start printing. If the printer cable is loose or is not connected to your Aquarius computer, simply connect the cable.

---

---

However, if you don't have an Aquarius printer or if the printer cable is loose, Logo waits patiently for a response. Logo does not return control to you until the list of procedures has been printed. In this instance, Logo does not respond to **CTL** or **G**. Warm start Logo to cancel the LP command and return to the command mode without erasing Logo's memory.

Similarly, when you use the LOAD command to load procedures and variables from cassette, Logo returns control to you **after** the last procedure or variable has been loaded. If the recorder is not connected, simply connect the recorder. If you don't have a recorder, warm start Logo to save Logo's memory and return to the command mode.

When Logo waits for a nonexistent printer to respond, Logo is said to "hang." That is, Logo only responds to the **RST** key.

In other instances, for example while running a program, Logo can also appear to hang. If **CTL** or **G** doesn't return control to you, try a warm start.

Sometimes, Logo becomes so confused that Logo doesn't know where or how to continue. When this happens, Logo "crashes," and loses its memory. A crash is usually indicated by the message I DON'T UNDERSTAND 255. Or a crash may be more dramatic: your program is terminated, and you return to Logo's title page.

If you think that Logo has crashed, try warm starting Logo first. In the command mode, use the NAMES and P commands to find out whether anything has been lost. If a warm start does not return you to the command mode, cold start Logo by turning the computer off and then back on.

---

## REPRESENTATION OF NUMBERS

Numbers are represented either with or without a decimal point. Numbers represented without a decimal point are called “integers.” For example, 1, 2, and 3 are integers.

Numbers that are represented with a decimal point are called “floating-point numbers.” For example, 12.8 and 59.2 are floating-point numbers.

The left-most digits in a number are called the “significant” digits, because these digits represent the significant — or most important — part of the number.

Programming languages like Logo have to decide how many digits they should represent precisely in numbers. Logo, for example, uses six-digit precision. That is, Logo only keeps the first six significant digits in a number.

Normally, you don't use numbers containing six or more digits in Logo programs. For example, you normally work with relatively small numbers that represent the position of turtles as x,y coordinates.

However, when you use more than six digits, you should be aware how Logo — and other programming languages — represent numbers.

For example, try printing the value 123456789:

```
? PRINT 123456789  
1.23457E+08
```

---

Since it would be misleading to only print the first six digits in this number, Logo represents the number in a different format called "scientific notation." Notice that the first six significant digits are shown, and the last digit is rounded up.

The notation  $E + 08$  means that position of the decimal point is really 8 places to the right. Similarly, the notation  $E - 08$  would indicate that the position of the decimal point is really 8 places to the left.

Normally, you do not encounter scientific notation in Logo programs.

## SPECIAL CHARACTERS IN NUMBERS

How would you enter \$1,500.99? If you were writing a check or doing your math homework, the dollar sign and comma would be meaningful to you. Logo, on the other hand, only understands a leading plus or minus sign, the digits zero through nine, the decimal point, and the letter E in scientific notation. Always enter at least one digit on both sides of the decimal point. For example, enter 5. as 5.0 and .5 as 0.5.

Enter PRINT \$1,500.99. What happens?

```
? PRINT $1,500.99
```

```
I DON'T KNOW HOW TO $1,500.99
```

Now, try PRINT 1500.00. What happens?

```
? PRINT 1500.99
```

```
1500.99
```

---

---

## OPERATORS

Logo uses numeric and relational operators. These operators are evaluated in a default order unless you use parentheses to change the order of evaluation.

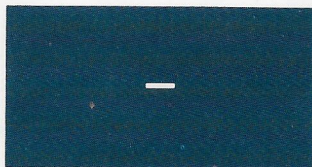
### NUMERIC OPERATORS



Use the plus sign for addition. The value to the right of the plus sign is added to the value to the left of the plus sign. Spaces to the left or right of the plus sign are not important.

```
? PRINT 5 + 6  
11
```

Do not use plus signs to indicate that numbers are positive. Unless a number is preceded by a minus sign, Logo assumes that a number is positive. For example, `PRINT 4 + 3` returns 7, but `PRINT + 4 + 3` returns I DON'T UNDERSTAND 5.



Use the minus sign for subtraction. The value to the right of the minus sign is subtracted from the value to the left of the minus sign. Spaces to the left or right of the minus sign are not important.

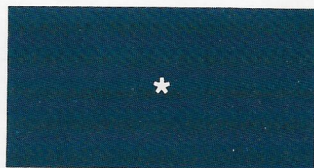
? PRINT 5 - 2

3

The minus sign can also be used to form the negative of a number. Use one or more spaces in front of the minus sign, but do not use spaces between the minus sign and the number.

? MAKE "X -2

-2



Use the asterisk for multiplication. The numbers on either side of the asterisk are multiplied together. Spaces to the left or right of the asterisk are not important.

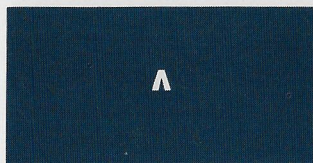
? PRINT 5\*2

10



Use the slash for division. The number to the left of the slash is divided by the number to the right. Spaces to the left or right of the slash are not important. Note that Logo does not divide by zero.

```
? PRINT 1/3  
.333333
```



Use the caret for exponentiation. For example,  $2 \wedge 5$  causes 2 to be multiplied by itself 5 times. Spaces to the left or right of the caret are not important.

```
? PRINT 2 ^ 5  
32
```

## RELATIONAL OPERATORS

Relational operators evaluate two values and return true (T) or false (NIL).

Operator	Description	Example
<	Less than	:A < :B
>	Greater than	:A > :B
=	Equals	:A = :B
<=	Less than or equal to	:A <= :B
>=	Greater than or equal to	:A >= :B
<>	Not equal to	:A <> :B

---

## OPERATOR PRECEDENCE

Arithmetic operations are performed in a default order. From left to right, Logo performs arithmetic in the following order:

Parentheses ( :A + :B )

Exponentiation :A ^ :B

Negation (assigning a number a negative value) MAKE "A -1

Multiplication or division :A \* :B or :A / :B

Addition or subtraction :A + :B or :A - :B

Relational operations :A > :B :A = :B

## MEMORY REQUIREMENTS

Logo divides memory into two parts: your workspace and the edit buffer. The smallest unit of storage is called a "byte." Each letter, digit, and special character requires the same amount of space in memory: one byte. Memory is also measured in K increments of 1024 bytes. For example, a 4K RAM cartridge provides 4096 bytes of memory.

## WORKSPACE

Your workspace is used to store the words, lists, variables, strings, and procedures that you define. Use the FREE command to find out how many bytes of workspace are available to you. The amount of available space depends on whether you are using a RAM cartridge, and the number and size of procedures that are currently stored in memory.

---

---

## EDIT BUFFER

The edit buffer, on the other hand, is a separate portion of memory that Logo uses when you enter the EDIT command. The size of the edit buffer is always 1K, even when you use a RAM cartridge.

**NOTE:** Do not create procedures in the edit mode unless you are using a RAM cartridge.

## WORDS AND STRINGS

In the procedure definition mode and edit mode, the length of words and strings is limited by the line length. When you use a 16K RAM cartridge, however, you can create words and strings up to 89 characters in length through list processing.

List processing allows you to create long words and strings by continually adding a word or string to an existing word or string. Forming words or strings by joining characters in this manner is called "concatenation."

### Concatenating Words

The following program creates a very long name by repetitively adding the word A to the current value of variable A.

```
TO BIG
  MAKE "A WORD :A "A
  MAKE "C :C + 1
  PRINT :A
  PRINT :C
  BIG
END
```

---

To run this program, enter the following lines:

```
? MAKE 'A 'A RTN
? MAKE 'C 1 RTN
? BIG RTN
```

In response, Logo prints the contents of the word and the number of characters in the word for each iteration of the loop.

```
AA 2 AAA 3 AAAA 4 ...
```

If you are using a 16K RAM cartridge, the program terminates in the message I DON'T UNDERSTAND 13 when Logo attempts to form a word containing 90 characters.

Due to tighter operating parameters when you use less RAM, the maximum length of words formed through list processing is smaller, and the same program terminates with the message I DON'T UNDERSTAND 255. Warm start Logo to continue.

With a 4K cartridge, the maximum length of words formed through list processing is variable — approximately 58 characters. Similarly, when you do not use a RAM cartridge, the maximum length of words formed through list processing is also variable — approximately 38 characters.

### Concatenating Strings

To create a long string instead of a long word, change the first line of this program from

```
MAKE 'A WORD :A 'A
```

to

```
MAKE 'A STRING :A 'A
```

---

---

Then, enter the following lines to run the program:

```
? MAKE 'A 'A' RTN  
? MAKE 'C 1 RTN  
? BIG RTN
```

In response, Logo prints the contents of the string and the number of characters in the string for each iteration of the loop.

```
'AA' 2 'AAA' 3 'AAAA' 4 ...
```

If you are using a 16K RAM cartridge, the program terminates in the message I DON'T UNDERSTAND 13 when Logo attempts to form a string containing 90 characters.

Due to tighter operating parameters when you use less RAM, the maximum length of strings formed through list processing is smaller, and the same program terminates with the message I DON'T UNDERSTAND 255. Warm start Logo to continue.

With a 4K cartridge, the maximum length of strings formed through list processing is variable — approximately 72 characters. Similarly, when you do not use a RAM cartridge, the maximum length of strings formed through list processing is also variable — approximately 38 characters.

---

## LISTS

Unlike words and strings, space for lists is allocated, as needed, in blocks of 256 bytes, called "pages." The maximum length of a list is determined by the amount of unused workspace currently available to you.

Obviously, more workspace is available when you use a 4K or 16K RAM cartridge. However, workspace is also used to store words, strings, procedures, variables, and other lists. The amount of space actually available to you is the remainder of the space not currently being used to store information.

There are two types of lists: long lists and deeply nested lists:

Long List

[ A A A A ]

Deeply Nested List

[ A [ A [ A [ A ] ] ] ]

Long Lists

Use the following program to create a very long list by repetitively adding the name A to a list:

```
TO BIG
  MAKE "A LPUT "A :A
  MAKE "C :C + 1
  PRINT :A
  PRINT :C
  BIG
END
```

---

To run this program, enter the following lines:

```
? MAKE "A [ A ] RTN
? MAKE "C 1 RTN
? BIG RTN
```

In response, Logo prints the contents of the list and the number of characters in the list for each iteration of the loop.

```
[ A A ] 2 [ A A A ] 3 [ A A A A ] 4 ...
```

Eventually, this type of program uses up all of the available memory on your system, and terminates in the NO ROOM message. In addition, since Logo requires a certain amount of memory to process commands, each command that you enter thereafter may terminate in NO ROOM. (See Coping with NO ROOM.)

### Deeply Nested Lists

To create a deeply nested list, change

```
MAKE "A LPUT "A :A
```

to

```
MAKE "A LPUT :A [ A ]
```

Then, enter the following lines to run the program:

```
? MAKE "A [ A ] RTN
? MAKE "C 1 RTN
? BIG RTN
```

---

---

Eventually, this type of program uses up all of the available memory on your system, terminates in the message I DON'T UNDERSTAND 255, and halts. In this example, Logo is not able to recover memory after the program terminates. Logo displays the message I DON'T UNDERSTAND 255 after each warm start. To continue, turn the console off, and start over. (See "Coping with NO ROOM".)

## STACKS

A "stack" is a special kind of list that Logo uses to keep track of where processing continues after a procedure has been executed.

For example, suppose that procedure A calls procedure B. After procedure B has been executed, Logo continues at the statement following the call to procedure B. The location of this statement appears at the beginning of the list.

TO PROCA

.  
.

PROCB

next statement Processing continues here after PROCB has finished.

.  
.

END

When procedures are nested (that is, when procedure A calls procedure B, which in turn calls procedure C), these return locations are pushed down in the stack in the order in which the procedures are called. In this manner, the return address for the last procedure called always appears at the beginning of the list.

---

---

Logo displays the NO ROOM message when you run out of stack space. Although stack space is allocated in proportion to the amount of RAM in use, this condition usually indicates that you are calling procedures improperly — even when you use a 16K RAM cartridge.

For example, the following program uses up stack space very quickly, and terminates in the NO ROOM message:

```
PROC1  
  PROC2  
END
```

```
PROC2  
  PROC1  
END
```

The same type of problem occurs when you use CATCH and THROW improperly. For example, CATCH executes procedure PUT1 until the turtle goes out of bounds. Then, processing continues at the statement following CATCH — in this example THROW “ERROR. However, the THROW statement recursively calls the procedure in which THROW appears. CATCH and THROW are not designed to handle this type of transfer of control.

```
TO PUT  
  CATCH “ERROR [ PUT1 ]  
  THROW “ERROR [ PUT ]  
END
```

```
TO PUT1  
  FD 1 PUT1  
END
```

---

---

## PROGRAMMING TIPS FOR USING RECURSION

“Recursion” refers to the ability of a procedure to call itself. Typically, the recursive statement is the last statement in a procedure.

Procedures that contain a recursive statement at the end of the procedure are called “tail-recursive procedures.” For example, the following procedure is tail-recursive:

```
TO FLY :S
  :S < 16
  IFTRUE [ FD :S RT 90 FLY :S + 1 ]
END
```

To run this procedure, type HOME CS HD 90, then type FLY 1, and press RTN.

Although recursive procedures need not be tail-recursive, tail recursion is preferred, because this programming technique uses very little stack space.

For example, you could write the same program as follows:

```
TO FLY :S
  :S < 16
  IFTRUE [ FLY :S + 1 ]
  FD :S RT 90
END
```

---

---

Try running this program without external RAM. After you create the program, type HOME CS HD 90, then type FLY 9 and press RTN. Notice how the sides of the figure **start** with a length of 16 and diminish by one down to nine.

Now, type FLY 1 and press RTN.

Logo returns NO ROOM. Why? Logo procedures that are not tail-recursive use one location in the stack for each recursive call. Then, the statements following each recursive call are executed in inverse order. In this example, the first value of :S is 16, the next is 15, and so on.

When you use a 4K or 16K cartridge, you can create interesting effects. For example, type POS [ 0 59 ] CS HD 90. Then type FLY - 16 and press RTN. Logo draws the figure counterclockwise starting with a value of 16, then draws the figure clockwise from 0 to - 16. Remember that FD - 16 is the same as BK 16.

## COPING WITH NO ROOM

NO ROOM is intended to be an advisory message, like the gas gauge in the family car. When the needle gets close to EMPTY, you can probably drive around for quite a while, but it would be better to put some gas in the tank.

Similarly, when Logo displays the NO ROOM message, you may be completely out of memory, or you may have enough memory to use commands like NAMES, SAVE, and ER.

### Arithmetic

The FACE command and arithmetic operations involving floating-point arithmetic temporarily require a page of memory (256 bytes) to calculate a result. When you enter PRINT 1/3, you are actually asking Logo to find an empty page of memory to perform the arithmetic. If the command terminates in NO ROOM, Logo could not find an empty page of memory. Usually, you can perform a number of other operations before Logo displays NO ROOM again and really means it.

---

---

## FREE Command

After NO ROOM is displayed, most people enter the FREE command to find out exactly how close they really are to running out of memory. In some cases, the number that FREE returns may be misleading.

The FREE command is intended to return the number of **unused** bytes of memory. This byte count includes the unused bytes in pages of memory allocated to lists. However, those unused bytes are not available for other purposes.

## Command Processing

Sometimes, there may be enough memory to initiate a command, but not enough memory to complete the command. For example, a certain amount of memory is required for the lists that commands like NAMES and TF generate.

If you are completely out of memory, each command that you enter terminates in NO ROOM, since Logo requires a certain amount of memory to process commands. To continue, turn the console off and start over.

## Procedure Definition

If NO ROOM is displayed while you are using the TO command to define a new procedure, NO ROOM means that you are completely out of memory. Logo cannot save the procedure. To avoid this problem, use additional RAM.

## Edit Mode

When you enter the Q command to exit the edit mode, NO ROOM indicates that Logo could not find room for the new or revised procedure definition. Try entering the Q command again. If NO ROOM is displayed again, warm start Logo. You lose the current procedure definition, since Logo cannot find space to save it. Use additional RAM to increase Logo's memory.

---

---

## Stack Overflow

NO ROOM may also indicate that you are calling procedures improperly. For example, procedure A and procedure B call each other until the stack overflows.

```
TO PROC A
  PROC B
END
```

```
TO PROC B
  PROC A
END
```

The same type of problem may occur when you use CATCH and THROW improperly, or when you use recursion in a manner that uses stack space excessively. (See “Stacks” and “Programming Tips for Using Recursion”).

---

---

## TURTLE PROGRAMMING TIPS

When you hatch a turtle, the turtle assumes the properties of the current turtle, including position and shape. Once hatched, turtles are not displayed until you move them, or until you change their shape.

For example, after a cold start, you hatch turtles Andy and Fred. These turtles are not displayed on the screen until they are moved:

```
? HATCH "ANDY  
ANDY  
? HATCH "FRED  
FRED  
? ASK :STUDS W 10  
? ASK :ANDY N 10  
? ASK :FRED S 10
```

When you move turtles to the same screen position, the last turtle moved to that position is displayed on the screen. Remaining turtles in that position are not displayed until you move them or until you change their shape.

---

---

## PLOTTING TURTLE TRACKS

The Aquarius Logo character set contains letters, numbers, and special characters. Some of the special characters are called “block characters.”

Block characters are intended to be used to draw geometric figures and create special effects. Each block character occupies the same space as a number, letter, or other special character.

Each block character is composed of six blocks, any of which may be filled or blank. For example, characters 241, 242, and 255 are block characters.

The tiles that a turtle plots replace the existing tiles in a given space on the screen. The only exception is the default tile, character 255.

The default tile does not replace letters or special characters. This is why the turtle appears to skip over these characters as though the turtle were in the pen up state.

For example:

```
? HOME
? CS
? PEN 1
? TILE 65
? W 1Ø
? TILE 255
? E 1Ø
```

However, unlike other tiles, the default tile forms “composite characters” with other block character tiles.

---

---

For **example**, tile 243 is formed when the default tile crosses tile 241 or tile 242:

```
? HOME
? CS
? TILE 241
? W 10
? TILE 255
? E 10
? TILE 242
? W 10
? TILE 255
? E 10
```

## ERASING TURTLE TRACKS

The **Aquarius** Logo character set contains two blank characters: 32 and 160. Character 32 is the text **blank**, and character 160 is the block character blank. Use character 160 to erase turtle tracks. Although you can use character 32 to erase turtle tracks, block character turtle tracks skip over these **spaces**.

For **example**, the following commands illustrate how default turtle tracks skip over character 32 turtle **tracks** as though the turtle were in the pen up state. The reason for this is that the default tile is a block **character**, and block characters do not replace nonblock characters.

```
? HOME
? CS
? PEN 1
? TILE 255
? W 10
```

---

---

? TILE 32

? E 1Ø

? TILE 255

? W 1Ø

Now, try using character 16Ø instead of character 32 to erase the turtle tracks. This time, the turtle tracks are plotted when you cross over the blank tiles:

? HOME

? CS

? PEN 1

? TILE 255

? W 1Ø

? TILE 16Ø

? E 1Ø

? TILE 255

? W 1Ø

## WHAT TO DO WHEN THE TURTLE WON'T BUDGE

The 6Ø by 8Ø graphics screen is divided into 2Ø by 4Ø tile spaces starting in the upper left-hand corner of the screen. Each tile space is divided into three horizontal rows by two vertical columns.

When you use the default tile shape, a dot is placed in one of these positions in a tile space each time the turtle moves one unit in the pen down state.

---

---

In contrast, when you use one of the other tile shapes, the turtle has to move to the edge of a tile space before the tile appears on the screen. For example:

? HOME

? CS

? PEN 1

? TILE 212

? N 1

*When you move north from the home position one unit, you coincidentally move to the edge of a tile space. The tile shape pops up on the screen.*

? N 1

*You move north one unit, but the tile shape does not appear on the screen.*

? N 1

*You move north one unit again. Again, the tile shape does not appear on the screen.*

? DOT

*You try to place a dot in the turtle's current position, but nothing happens, since you have not moved to the edge of the tile space yet.*

? N 1

*After moving north three units, you reach the edge of the tile space, and the tile shape is placed on the screen.*

? E 1

*The same thing happens when you move east one unit: the tile*

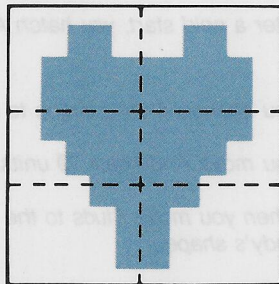
? E 1

*shape is not displayed until you move east again.*

---

---

Each tile space is divided into three horizontal by two vertical parts.



When the turtle starts from this position, the turtle has to move north three units or east two units before a nondefault tile shape appears on the screen.

## TURTLE SHADOWS

Each time the turtle moves, the turtle replaces the shape in the new screen position. But, turtles remember the shape that they replace. Then, when a turtle moves, the turtle restores the shape that it remembered in that position.

Turtles cannot tell whether a shape is the shape of a tile or the shape of a turtle.

When you move turtles to the same position on the screen, you should move the turtles away in the opposite order in which you originally moved them to that position. Why? If you move the turtles in a different order, you may end up with "turtle shadows."

A turtle shadow is the shape of a turtle that another turtle restores to a screen position even though the previous turtle has already moved.

---

---

For example:

? HATCH "ANDY  
ANDY

*After a cold start, you hatch Andy.*

? ASK :ANDY SHAPE 212

*You change Andy's shape to a heart.*

? E 10

*You move Andy east 10 units.*

? ASK :STUDS E 10

*When you move Studs to the same position, Studs remembers Andy's shape.*

? ASK :ANDY W 10

*You move Andy in the wrong order.*

? ASK :STUDS S 10

*When you move Studs, Studs leaves Andy's shadow behind.*

In this example, you should have moved Studs before you moved Andy. Use the CS or SS command to remove turtle shadows.

---

## APPENDIX B

---

### LOGO COMMAND SUMMARY

Commands like INK, and operators like + and −, are often called “primitives.” These keywords and symbols are the basic or fundamental building blocks of the Logo language. When you define procedures and variables, the procedure names and variable names are added to Logo’s vocabulary.

In this reference section, Logo commands are arranged alphabetically by command name. Some of the examples in this reference section require more memory than is built into your Aquarius Home Computer System. Use a 4K or 16K RAM cartridge in the Aquarius Mini Expander to expand Logo’s memory.

#### ALL [ relation ... ]

ALL tests a list of relational expressions. If each of the expressions is true, ALL returns the word T (true). Otherwise, if one or more expressions are false, ALL returns the word NIL (not true).

In the immediate mode, ALL displays the word T or NIL after evaluating each list of relational expressions.

```
ALL [ 5=2+3 6>4 ]
```

```
T
```

```
ALL [ 3+3>7 ]
```

```
NIL
```

---

## ANY [ relation ... ]

ANY tests a list of relational expressions. If at least one of the expressions is true, ANY returns the word T (true). Otherwise, if none of the expressions is true, ANY returns the word NIL (not true).

1. In the immediate mode, ANY displays the word T or NIL after evaluating each list of relational expressions.

```
? ANY [5=2+3 6>4]
```

```
T
```

```
? ANY [3+3>7 4-5>0]
```

```
NIL
```

```
? ANY [6>4 3+3>7]
```

```
T
```

2. The following program draws squares in random colors and turtle tracks from random positions on the screen. In order to keep the figures from going out of bounds, you use ANY to ask whether drawing a square from a given position would cause the figure to go out of bounds. If the answer is yes, you calculate the longest allowable width of the square before the square is drawn.

Notice how you recursively use procedure BLKSIZ to reduce the size of the figure to fit. Also notice how you assign the final size of the figure (:VAR) to variable :SIZE in BLKSIZ. This technique allows you to return the value of the variable to the calling procedure. Execute LEAPS to run the program (requires 4K/16K RAM cartridge).

```

TO LEAPS
  HD 90
  PEN 0
  INK RN 16
  TILE RN 256
  POS ( LIST RN 80 RN 60 )
  MAKE "PX FIRST POS ?
  MAKE "PY LAST POS ?
  MAKE "VAR 20
  BLKSIZ :VAR
  PEN 1
  REP 4 [ FD :SIZE RT 90 ]
  LEAPS
END

```

```

TO BLKSIZ :VAR
  ANY [:PX + :VAR > 79 :PY - :VAR < 0 ]
  IFFALSE [ MAKE "SIZE :VAR ]
  IFTRUE [ BLKSIZ :VAR-1 ]
END

```

**ASK :turtle**  
or  
**ASK ?**

ASK takes a single input: either the name of a turtle or a question mark. When the input is the name of a turtle, the turtle becomes the current turtle. Subsequent commands talk to that turtle until you make another turtle the current turtle. When the input is a question mark instead of a turtle name, ASK returns the name of the current turtle.

Use HATCH to bring new turtles into the turtle family, use TF to display the names of the turtles currently in the turtle family, and use ER to remove selected turtles from the turtle family. Note that Logo does not erase the current turtle.

? ASK ?  
FRED

*You ask who the current turtle is. Logo returns the name of the current turtle. In this example, FRED is the current turtle.*

? HATCH "JIM  
JIM

*You hatch a new turtle called JIM. Logo responds by displaying the new turtle name.*

? ASK :JIM  
FRED

*You make JIM the new current turtle. Logo prints the name of the former current turtle.*

? (ASK ?) = :JIM  
T

*You ask if JIM is the current turtle. Logo says "yes" (that is, Logo returns a T for "true").*

**B**

B moves the turtle backwards 10 units on the current heading.

The following example draws a line to the right 10 units and then down 10 units. Notice that the B command draws a line in the opposite direction of the current heading without changing the heading. After you enter the B command, the heading is still 90 degrees.

```
? HOME CS
```

```
? E 10
```

```
? B
```

**BF [ list ]**  
or  
**BF 'string'**

BF (stands for BUT FIRST) returns all but the first element in a list, or all but the first character in a string.

```
? BF [ A B C D ]
```

```
[ B C D ]
```

```
? BF 'abcd'
```

```
'bcd'
```

```
? BF (POS ?)
```

```
[ 30 ]
```

## BK number

BK (stands for BACK) moves the turtle backwards on the current heading.

Numbers containing a decimal fraction are truncated to integer values (that is, BK 10.6 has the same effect as BK 10). Also, negative numbers make BK act like FD. For example, BK -8 is equivalent to FD 8.

The following example draws a line to the right 10 units and then down 10 units. Notice that BK draws a line in the opposite direction of the current heading without changing the heading. After you enter the BK command, the heading is still 90 degrees.

```
? HOME CS
```

```
? E 10
```

```
? BK 10
```

BL [list]  
or  
BL 'string'

BL (stands for BUT LAST) returns all but the last element in a list, or all but the last character in a string.

```
? BL [ A B C D ]
```

```
[ A B C ]
```

```
? BL 'abcd'
```

```
'abc'
```

```
? BL (POS ?)
```

```
[ 40 ]
```

## CATCH “name [command ... procname ]

CATCH is used in conjunction with THROW to perform unusual transfers of control. Typically, you use CATCH to catch error conditions before those errors terminate your program. You can also use CATCH and THROW to define and catch program-specific

error conditions such as variables too large or too small.

CATCH uses two inputs: a label that identifies corresponding THROW statements, and a list of commands. Commands in the list are executed in the order in which the commands appear in the list — until a THROW statement with the same label is encountered during program execution. Then, the commands in the THROW statement are executed.

Each THROW statement may transfer control to one or more procedures, and multiple THROW statements having the same label may be executed. Control eventually returns to the statement following CATCH. THROW statements may not be executed until their corresponding CATCH is executed (refer to description of THROW).

Procedure names and commands following the first procedure name in the CATCH command list are ignored.

**1.** Internally, when an error occurs, Logo executes a THROW “ERROR statement that returns you to Logo’s command level. To intercept this error, you can use a CATCH “ERROR statement as shown in the following program.

To run this program, execute procedure PUT. PUT advances the turtle to the top of the screen until an out-of-bounds error occurs. Then, PUT returns the turtle to the home position and starts over.

Notice how PUT uses the CATCH statement to call PUT1 until the out-of-bounds error occurs. When the error occurs, Logo internally issues a THROW "ERROR statement. According to the rules of CATCH and THROW, control returns to the statement following CATCH.

```
TO PUT
  CATCH "ERROR [ PUT1 ]
  HOME
  CS
  PUT
END
```

```
TO PUT1
  FD 1
  PUT1
END
```

**2.** Suppose that you wanted to catch certain program-specific error conditions, generate your own error or warning message, and continue. Modify PUT1 as shown. Then, run PUT as before.

```

TO PUT1
  FD 1
  ( LAST POS ? ) = Ø
  IFTRUE [ THROW "ERROR [ PRINTNL
    'Your own error message' ] ]
  PUT1
END

```

**3. RNDBLOCKS** draws blocks in random colors and random turtle tracks. Whenever the turtle goes out of bounds, CATCH catches the error. Then, the procedure continues through recursion instead of terminating with an error message.

```

TO RNDBLOCKS
  PEN Ø
  POS ( LIST RN 8Ø RN 6Ø )
  PEN 1
  INK RN 16
  TILE RN 256
  MAKE "X RN 2Ø
  CATCH "ERROR [ N :X W :X S :X E :X ]
  RNDBLOCKS
END

```

**CH 'character'**  
or  
**CH number**

CH returns the character code for a one-character string, or the character represented by a given character code. When a number over 255 is used, 256 is subtracted from the value until the remainder is less than 256.

? CH 'k'

107

? CH 107

'k'

**CHECK 'filename' [procname ... ]**  
or  
**CHECK 'filename' NAMES**

CHECK verifies whether procedures have been properly saved on cassette, or properly loaded into memory. The procedure names you identify in CHECK must appear in the same order in which the procedure names appeared in the corresponding

SAVE command. Use the function NAMES in the CHECK command if you used the function NAMES to save the procedures.

CHECK skips over cassette files until it finds the identified file. Each time CHECK skips over a file, CHECK prints a #. Remember that Logo distinguishes between lowercase and uppercase in filenames. (See Chapter 7 for details.)

SAVE 'TEST' [ LEAPS HOUSE SPIRAL ]

CHECK 'TEST' [ LEAPS HOUSE SPIRAL ]

SAVE 'OCT25' NAMES

CHECK 'OCT25' NAMES

**COND [[ relation action ... ]**

**[ relation action ... ]]**

COND allows you to conditionally execute lists of commands. Each relational expression is evaluated sequentially starting from the beginning of the list. Only one list of commands is executed: the commands associated with the first true expression are

executed. The remaining relational expressions and their associated lists of commands are ignored.

The following procedure generates random X coordinates in the range 0-79 and random Y coordinates in the range 0-59. COND sets the heading and places a dot based on the value of the X coordinate (requires 4K/16K RAM cartridge).

TO TURN

HOME CS TURTLE 0 PEN 0

POS (LIST RN 80 RN 60)

MAKE "PX FIRST POS ?

PRINTNL POS ?

COND [ [ :PX <= 20 HD 0 DOT ]

[ :PX >= 60 HD 180 DOT ]

[ :PX <= 45 HD 270 DOT ]

[ :PX <= 60 HD 90 DOT ] ]

PRINTNL HD ?

END

---

CS

CS (stands for CLEAR SCREEN) erases the turtle tracks from the Graphics Screen without erasing the turtles or changing their properties.

---

DELAY

DELAY causes Logo to pause slightly before executing the next statement. Use additional DELAY statements to create longer delays.

This program draws a “squirrel” until the turtle goes out of bounds. To run this program, enter CS, HOME, and then DRAW 1. Notice how DELAY slows down the animation pace.

```
TO DRAW :SIZE  
  DELAY  
  FD :SIZE  
  RT 90  
  DRAW :SIZE+1  
END
```

**DOT**

DOT places a dot in the turtle's current position — even in the pen up (PEN  $\emptyset$ ) state. Use the TILE command to change the shape of the dot.

? PEN  $\emptyset$

? TURTLE  $\emptyset$

? FD 1 $\emptyset$

? DOT

? FD 1 $\emptyset$

? DOT

**E number**

E (stands for EAST) moves the turtle a specified number of turtle units to the right without changing the turtle's current heading.

Numbers containing a decimal fraction are truncated to integer values (that is, E 1 $\emptyset$ .6 has the same effect as E 1 $\emptyset$ ). Negative numbers make E act like W. For example, E -8 is equivalent to W 8.

**EDIT procname**  
or  
**EDIT "procname"**

EDIT invokes Logo's edit mode. In the edit mode, you can add, change, or delete Logo commands in an existing procedure, or create a new procedure.

You use the area from the top of the screen down to the prompt line at the bottom of the screen. Use the TO line at the top of the screen to enter procedure inputs. If necessary, use additional lines for procedure inputs. Enter the first line of the procedure body on the next line. Do not enter Logo commands on the END line.

When you enter the Q command to exit the edit mode, the procedure is saved down to the first END statement. If your procedure contains multiple END statements, procedure lines following the first END statement are discarded.

A procedure cannot be edited unless the entire procedure fits on the edit screen. Divide a procedure that is too large to fit on the screen into smaller procedures.

**NOTE:** Optionally, the procedure name may be preceded by a quotation mark. If used, the leading quotation mark is not considered to be part of the procedure name.

---

## SCREEN REFORMATTING

When you enter the EDIT command to edit an existing procedure, Logo displays the procedure on the screen. If the procedure does not fit on the screen, Logo reformats the procedure until the procedure fits on the screen in its entirety. The procedure scrolls up on the screen each time the procedure is reformatted.

While being reformatted, a procedure may scroll two or more times. However, a procedure that is too large to fit on the screen scrolls continuously. If it is obvious to you that Logo cannot reformat the procedure after several tries, warm start Logo, and redefine the procedure as a group of smaller procedures.

If you are unsure whether a procedure will fit on the screen, define the procedure in the edit mode.

## POSITIONING THE CURSOR

The following keys may be used to position the cursor, except in the insert mode:

Key	Description
<b>E</b>	Move cursor up one line.
<b>S</b>	Move cursor left one space. When the cursor is at the beginning of a line, S moves the cursor to the end of the preceding line.
<b>D</b>	Move cursor right one space. When the cursor is at the end of a line, D moves the cursor to the beginning of the following line.
<b>X</b>	Move cursor down one line.

---

---

## INSERTING TEXT

To insert text, press the **I** key to initiate the insert mode. Then, type the text to be inserted. The characters that you type appear on the screen in the cursor position. Each time you type a character, the cursor and the remainder of the line moves one space to the right — up to the end of the current line.

When the current line is filled, the buzzer rings, and text insertion is inhibited. Press **RTN** to continue. Each time you press **RTN** the cursor and the characters from the cursor position to the end of the line move to the beginning of a new line.

To terminate the insert mode, type a semicolon (;).

**NOTE:** Do not use **RTN** to divide numbers, words, or strings. If the object does not fit in the remainder of the current line, move the object to the beginning of the next line.

## DELETING TEXT

In the insert mode, press the left-arrow key or **CTL-H** to delete the character to the left of the cursor. When the cursor is at the beginning of a line, pressing the left-arrow key or **CTL-H** deletes the character at the end of the preceding line.

When you are not in the insert mode, press **Z** to delete the character in the cursor position, or press **K** to delete the line in which the cursor appears.

## TERMINATING THE EDIT MODE

You can either terminate the edit mode normally and save editing changes, or you can abort the edit mode and discard the changes.

To save editing changes and return to the Logo command mode normally, terminate the insert mode. Then, type **Q** and press **RTN**. The revised procedure definition replaces the procedure that was in effect before you entered the edit mode.

---

---

To discard changes you have made and return to the Logo command mode, press **RST** and then press **CTL-C** (warm start). When you return to the command mode, the current procedure definition is the one in effect before you entered the edit mode.

**NOTE:** Do not warm start Logo to cancel the Q command before you return to the Logo command mode. You will very likely lose the procedure definition, and you may lose all of your procedures.

## NO ROOM WHEN YOU QUIT EDIT MODE

When you enter the Q command, Logo deletes the old version of your procedure before Logo makes a copy of the new procedure in your workspace.

If the NO ROOM message is displayed when you try to exit the edit mode normally, try entering the Q command again. Sometimes, Logo can clear enough workspace to save the new version of your procedure. If Logo displays the NO ROOM message again, it means that Logo couldn't find enough workspace for the new version of your procedure. Warm start Logo to erase your edit buffer without affecting the other procedures in your workspace.

## CREATING NEW PROCEDURES IN THE EDIT MODE

To create a new procedure in the edit mode, you can enter the procedure in its entirety, or you can create the procedure based on an existing procedure.

To create a new procedure in its entirety, enter a new procedure name in the EDIT command. Then, use the insert mode to create the body of the procedure, and exit normally.

To create a new procedure based on an existing procedure, enter the name of the existing procedure in the EDIT command. Then, in the edit mode, change the procedure name to the name of the new procedure. Add, change, and delete commands to complete the new procedure. Then, exit the edit mode normally. The new procedure definition is saved in your workspace, and the original procedure definition remains unchanged.

---

---

## LARGE PROCEDURES

Avoid using the TO command to create large procedures. Large procedures cannot be edited unless they fit on the edit screen.

If you are unsure whether a new procedure will fit on the edit screen, create the procedure in the edit mode. Then, you can quit the edit mode and save the procedure when you start running out of space on the screen. Later, you can edit the procedure into smaller procedures.

For example, you enter EDIT PROC to create procedure PROC in the edit mode. On the edit screen, Logo displays the TO and END lines. Initiate the insert mode to enter Logo commands between these two lines:

```
TO PROC  
END
```

Before you run out of space on the edit screen, quit the edit mode, and save the procedure.

To divide the procedure into two smaller procedures, enter EDIT PROC to display the large procedure on the edit screen. Then, change the procedure name to a new procedure name, for example PROC1. Delete the lines that you do not want in PROC1, and quit the edit mode.

When you return to the command mode, procedure PROC is unchanged, but the new, smaller procedure is saved in your workspace.

Repeat this process for each new procedure that you want to edit out of the large procedure. After editing the last small procedure out of the large procedure, use the ER command to delete the large procedure.

This approach also works when you want to create a group of procedures that are similar to another procedure.

---

---

## RESERVED WORD PROCEDURE NAMES

If you inadvertently use a reserved word as a procedure name, the message I DON'T UNDERSTAND 4 is momentarily displayed at the bottom of the edit screen when you try to exit the edit mode normally. Change the procedure name to a name that is not a reserved word and does not duplicate the name of an existing procedure. Then, enter the Q command to save the new procedure definition.

## WRONG NAME

Sometimes, you inadvertently enter a new procedure name in the EDIT command when you really intended to edit an existing procedure. Logo thinks that you want to create a new procedure in the edit mode. Warm start Logo to discard the "new" procedure definition.

Similarly, if you identify an existing procedure that is not the procedure you meant to edit, warm start Logo to discard the "new" procedure definition, and try again. The original procedure definition remains unchanged in your workspace.

## USING DELIMITER CHARACTERS

The leading quotation mark at the beginning of a word and the leading apostrophe at the beginning of a string must begin on the same line as the word or string.

In contrast, parentheses and brackets used in lists need not appear on the same line as objects in the list.

---

For example, the following commands are equivalent:

```
POS ( LIST RN 80 RN 60 )      POS  
                                (  
                                LIST  
                                RN 80  
                                RN 60  
                                )
```

When used as delimiter characters, brackets and parentheses must be evenly paired. When you enter the Q command, Logo displays the message I DON'T UNDERSTAND 1 at the bottom of the screen. Logo does not allow you to exit until each delimiter parenthesis and bracket is paired with its corresponding bracket or parenthesis. Insert matching delimiter characters where required, and enter the Q command again.

By default, when you omit the closing apostrophe in strings, Logo inserts the delimiter character for you after you enter the Q command.

Strings may be terminated by an apostrophe or the end of the line.

## ER "name

ER (stands for ERASE) deletes the definition for a named procedure, variable, or turtle.

When you use ER to remove a turtle from the turtle family, the named turtle may not be the current turtle.

```
? TF
[ STUDS ANDY ]
```

```
? ASK ?
```

```
STUDS
```

```
? ASK :ANDY TURTLE Ø
```

```
? ASK :STUDS
```

```
ANDY
```

```
? ER "ANDY
```

```
ANDY
```

```
? TF
```

```
[ STUDS ]
```

```
? NAMES
```

```
[ HOUSE LEAPS S C STUDS ANDY ]
```

*Before removing Andy from the turtle family, you remove turtle Andy from the graphics screen, and then make Studs the current turtle.*

```
? ER "HOUSE
```

```
HOUSE
```

```
? ER "S
```

```
S
```

*You erase procedure HOUSE and variable S.*

```
? NAMES
```

```
[ LEAPS C STUDS ANDY ]
```

*You enter NAMES again to generate a list of current names defined in your workspace.*

**F**

F moves the turtle forward 10 units on its current heading. For example, REP 4 [ F RT 90 ] creates a box 10 units square.

**FACE [ column row ]**

FACE changes the turtle's heading to face a given X,Y coordinate. Note that FACE temporarily requires 256 bytes of workspace. Logo displays the NO ROOM message when it cannot find the required amount of space to execute FACE. Use a 4K/16K RAM cartridge to increase Logo's memory.

? HOME CS

? HD ?

90

? FACE [ 0 0 ]

? HD ?

144

**FD number**

FD (stands for FORWARD) moves the turtle forward a specified number of turtle units on the current heading.

Numbers containing a decimal fraction are truncated to integer values (that is, FD 10.6 has the same effect as FD 10). Also, negative numbers make FD act like BK. For example, FD -8 is equivalent to BK 8.

For example, REP 4 [ FD 10 RT 90 ] creates a box 10 units square.

**FIRST [ list ]  
or  
FIRST 'string'**

FIRST returns the first element in a list, or the first character in a string.

? FIRST (POS ?)

40

? FIRST [ A B C ]

A

? FIRST [ [ A B ] C ]

[ A B ]

? FIRST 'Aquarius'

'A'

**FPUT object [ list ]**

FPUT (stands for FIRST PUT) combines the elements of two inputs into a list. The elements of the first input are placed at the beginning of the list, and the elements of the second input are placed at the end of the list. The first input may be a number, string, word, or list. The second input must be a list.

? FPUT 10 [ A B C ]

[ 10 A B C ]

? FPUT "A [ B C ]

[ A B C ]

? FPUT 'A' [ 'quarius' ]

[ 'A' 'quarius' ]

## FREE

FREE displays the number of bytes of workspace available for defining new variables and procedures. Note that the amount of available workspace changes as you create and erase variables, and add, change, and delete procedures.

? FREE  
328

*You enter the FREE command.  
Logo responds that 328 bytes of  
workspace are available.*

## FS

FS turns the text window into a full-screen window. In the full-screen mode, the commands that you enter and the responses that Logo returns scroll up from the bottom of the screen. Turtle commands are ignored in the full-screen mode. Enter the SS command to return to the split-screen mode.

## HATCH "name

HATCH adds a new turtle to the turtle family. The properties of the hatched turtle are identical to the properties of the current turtle.

When you enter the HATCH command in the immediate mode, Logo echoes the turtle's name. Thereafter, you refer to the turtle as a variable. Use the ASK command to make a hatched turtle the current turtle.

? HATCH "ANDY

*You hatch a turtle named Andy.*

ANDY

*Logo echoes the turtle name.*

? ASK :ANDY

*You make ANDY the current turtle.*

## HD number or HD ?

HD sets the current turtle's heading, or returns the current turtle's heading. When the specified heading is greater than or equal to  $360$ ,  $360$  degrees is subtracted from the heading until the heading is less than  $360$ . That is, HD  $450$  is equivalent to HD  $90$  ( $450 - 360 = 90$ ), and HD  $360$  is equivalent to HD  $0$  ( $360 - 360 = 0$ ).

? HD 45

? HD ?

45

## HOME

HOME returns the current turtle to the "home position" (position  $40$ ,  $30$ ), and sets the heading to  $90$  degrees.

When PEN 1 is in effect, HOME causes the turtle track to be plotted from the turtle's current position to the home position. To prevent these points from being plotted, enter PEN  $0$  before you enter the HOME command. Or, enter CS to clear the screen after you enter HOME.

## IFFALSE [ command ... ]

IFFALSE executes a list of commands if the preceding condition is false. In the immediate mode, Logo automatically resets the condition to false after you press RTN.

```
? HD 180
? (HD ?) < 90 IFFALSE [ HD 90 ]

TO PROC
  HD 180
  (HD ?) < 90
  IFFALSE [ HD 90 ]
END
```

## IFTRUE [ command ... ]

IFTRUE executes a list of commands if the preceding condition is true. In the immediate mode, Logo automatically resets the condition to false after you press RTN.

```
? HD 180
? (HD ?) > 90 IFTRUE [ HD 90 ]

TO PROC
  HD 180
  (HD ?) > 90
  IFTRUE [ HD 90 ]
END
```

INK color  
or  
INK ?

INK sets the pen color for the turtle and the turtle track. Color codes range from 0 to 15. A question mark instead of a pen color returns the pen color code.

? INK 1  
? FD 10  
? INK ?  
1

INT number

INT (short for INTEGER) forms an integer by throwing away the fractional part of the number.

? INT 10.6  
10  
? INT -8.7  
8

L

L rotates the turtle's heading left 45 degrees. When the resulting heading is greater than or equal to 360, 360 degrees is subtracted from the heading until the heading is less than 360.

? HD ?  
90  
? L  
HD ?  
135

**LAST** [ list ]  
or  
**LAST** 'string'

LAST returns the last element of a list or string.

```
? LAST (POS ?)
30
? LAST [ A B C ]
C
LAST 'Aquarius'
's'
```

**LENGTH** [ list ]  
or  
**LENGTH** 'string'

LENGTH returns the number of elements in a list, or the number of characters in a string.

```
? LENGTH [ A B C ]
3
? LENGTH 'Aquarius'
8
? TF
[ STUDS ANDY ]
? LENGTH TF
2
```

**LIST** object ...

LIST creates a list of objects identified in the command. Objects can be numbers, strings, words, or lists. Enclose the LIST command and the list of objects in parentheses when there are more than two objects, or when another command references items in the generated list.

```
? LIST 'Aquarius' 'Logo'
[ 'Aquarius' 'Logo' ]

? LIST "Aquarius " "Logo
[ AQUARIUS LOGO ]
```

```
MAKE "TURTLES (LIST FIRST TF LAST TF)
[ STUDS ANDY ]
```

```
? POS (LIST RN 80 RN 60)
[ 24 32 ]
```

```
(LIST :A :0 :B)
[ APPLES ORANGES BANANAS ]
```

**LOAD 'filename'**  
or  
**LOAD ''**

Use the LOAD command to load a cassette file of Logo procedures and variables into memory. The definitions that you load replace the definitions currently in memory that have the same name. Other definitions currently in memory are not affected.

As an option, use two consecutive apostrophes, instead of a filename, to load the first Logo file encountered on cassette. The character # is displayed underneath the LOAD command each time a file or procedure is skipped. (See Chapter 7 for details.)

```
? LOAD 'TEST'
```

**LP "procname**  
or **LP [ procname ... ]**  
or **LP NAMES**

LP prints identified procedures on the Aquarius printer. Use the keyword NAMES to print all of the procedure definitions in memory. Procedure definitions scroll up on the screen as they are being printed. Use your Aquarius printer in the TEXT mode. Logo initializes the printer each time you enter the LP command.

? LP "HOUSE  
? LP [ HOUSE LEAPS RNDBLOCKS ]  
? LP NAMES

**LPUT object [ list ]**

LPUT (stands for LAST PUT) combines the elements of two inputs into a list. The elements of the first input are placed at the end of the list, and the elements of the second input are placed at the beginning of the list. The first input may be a number, string, word, or list. The second input must be a list.

? LPUT 10 [ A B C ]  
[ A B C 10 ]

? LPUT "A [ B C ]  
[ B C A ]

? LPUT 'quarius' [ 'A' ]  
[ 'A' 'quarius' ]

## LT number

LT (stands for LEFT TURN) rotates the turtle heading left. The input in degrees is added to the current heading. When the resulting heading is greater than or equal to 360, 360 degrees is subtracted from the heading until the heading is less than 360.

? HD ?

90

LT 45

? HD ?

135

? LT 300

? HD ?

75

## MAKE "name value

MAKE assigns a value to a named variable. The value may be a number, string, word, list, or the current value of a variable.

? MAKE "PX 10

10

*Assigns the value 10 to variable :PX.*

? MAKE "PX :PX + 1

11

*Assigns the variable :PX the current value of the variable, plus one.*

? MAKE "STUDS :ANDY

ANDY

*Assigns turtle Studs the same properties as turtle Andy.*

**MEMBER object [ list ]**  
or  
**MEMBER 'character' 'string'**

MEMBER determines whether a character appears in a string, or whether an object appears in a list. The object may be a number, string, word, or list.

If the object appears in the list, Logo returns a number indicating the position

of the object in the list. For example, the number 4 indicates that the object is the fourth element in the list, from left to right.

If the character appears in a string, Logo returns a number indicating the position of the character in the string. Remember that Logo counts each space as a position in strings, and distinguishes between lowercase and uppercase.

Logo returns the word NIL if the object does not appear in the list or if the character does not appear in the string.

? MEMBER "LOGO [ LOGO IS FUN ]

1

? MEMBER "LOGO [ [ LOGO IS ] FUN ]

NIL

? MEMBER 10 [ A B C 10 ]

4

? MEMBER 's' 'Logo is smart'

7

? MEMBER 's' 'LOGO IS SMART'

NIL

**N number**

N (stands for NORTH) moves the turtle a specified number of turtle units straight up without changing the turtle's current heading.

Numbers containing a decimal fraction are truncated to integer values (that is, N 10.6 has the same effect as N 10). Negative numbers make N act like S. For example, N -8 is equivalent to S 8.

**NAME? "name**

NAME? tells you whether a word is being used as the name of a variable or turtle. If the name is the name of a variable or turtle, NAME? returns T (true). Otherwise, NAME? returns NIL (false).

? NAME? "RNDBOXES  
NIL

? NAME? "STUDS  
T

? NAME? "PX  
T

## NAMES

NAMES returns a list of the procedures, variables, and turtles currently defined in memory. Use NAMES as a function in commands requiring a list of names.

? NAMES

*(displays the names of all of the turtles, variables, and procedures currently defined in memory)*

[ LEAPS PY PX STUDS ]

? LAST NAMES  
STUDS

*(displays the last object in the list generated by NAMES)*

? LP NAMES

*(prints procedure definitions for procedures currently in memory)*

? SAVE 'F99' NAMES

*(saves currently defined procedures on cassette under the file name F99)*

## NL

NL (stands for NEW LINE) brings the cursor down to the beginning of a new line.

? FS

? PRINTNL 'A' NL NL 'B'  
'A'

*(Logo inserts two lines of space.)*

'B'

**NTH [list] number**  
or  
**NTH 'string' number**

NTH returns an element from a list or a character from a string. The location of the element or character is identified positionally from left to right. When you use a floating-point number to identify the position, Logo ignores the digits to the right of the decimal point.

```
? NTH [ A B C ] 2
B
```

```
? NTH [ A [ B C ] D ] 2
[ B C ]
```

```
? NTH ' Logo ' 4
'g'
```

*(Remember that every space represents a character position to Logo.)*

```
? NTH 'Logo' 7
.'
```

```
? NTH [ A B C ] 5
NIL
```

```
? ASK NTH TF 2 POS ?
[ 40 30 ]
```

*You make the second turtle in the turtle family the current turtle, and you ask Logo to return the turtle's position.*

---

**P “procname**  
or  
**P [ procname ... ]**  
or  
**P NAMES**

P prints identified procedures in the text window. To use the full-screen window, enter the FS command before you enter P commands. Use the keyword NAMES to print all of the procedures currently defined in your workspace. Procedure definitions scroll up on the screen as they are being printed.

? P “HOUSE  
? P [HOUSE LEAPS RNDBLOCKS]  
? P NAMES

---

**PAPER number**  
or  
**PAPER ?**

PAPER sets the background color of the Graphics Screen. Color codes range from 0 to 15. A question mark instead of a color code identifies the current background color.

? PAPER 3  
? PAPER ?  
3

---

**PEN number**  
or  
**PEN ?**

PEN sets the pen state for the current turtle: the pen is either up or down. Use PEN 0 to set the pen up, or use PEN 1 to set the pen down. Turtle tracks are only shown when the pen is down. Use a question mark instead of a number to find out whether the pen is up or down.

? PEN 1  
? PEN ?  
1

---

```
? MAKE "DOWN Ø
? PEN :DOWN
? PEN ?
Ø
```

**POS [ number number ]**  
or  
**POS ?**

POS (stands for POSITION) moves the turtle to a point on the Graphics Screen. The first input identifies the X coordinate, and the second input identifies the Y coordinate. Logo terminates POS with an out-of-bounds error when the given coordinates are

beyond the Graphics Screen. Use a question mark instead of a list of X,Y coordinates to find out where the turtle is currently located on the screen.

```
? POS [ 2Ø 3Ø ]
? POS ?
[ 2Ø 3Ø ]
```

```
TO LEAP
  POS (LIST RN 8Ø
    RN 6Ø)
  TILE RN 256
  INK RN 16
  LEAP
END
```

*The procedure LEAP draws turtle tracks in random colors and characters to random positions on the screen. Run LEAP with the pen down (PEN 1). Then, press CTL - G to halt the program.*

---

## PRINT object

PRINT prints an object starting from the current cursor position on the command line. After the object is printed, the cursor remains to the right of the last character printed. An object may be a name, number, string, or list.

```
? PRINT 'Aquarius'  
'Aquarius' ?
```

```
? PRINT [ A B C ]  
[ A B C ] ?
```

```
? PRINT "LEAPS  
LEAPS ?
```

---

## PRINTNL object

PRINTNL prints an object starting from the current cursor position on the command line. After the object is printed, the cursor advances to the beginning of the next line. An object may be a name, number, string, or list.

```
? PRINTNL 'Aquarius'  
'Aquarius'  
?
```

```
? PRINTNL [ A B C ]  
[ A B C ]  
?
```

```
? PRINTNL "LEAPS  
LEAPS  
?
```

**PRINTSE object**

PRINTSE prints an object starting from the current cursor position on the command line. An object may be a name, number, string, or list. If the object is a list, the list is printed without the outermost brackets. After the object is printed, the cursor remains to the right of the last character printed.

? PRINTSE [ A B C ]

A B C ?

? PRINTSE POS ?

40 30 ?

? PRINTSE [ A [ B C ] D ]

A [ B C ] D ?

**R**

R rotates the turtle heading right 45 degrees.

When the new heading is less than or equal to zero, 360 is added to the heading until the heading falls in the range 0-360 degrees.

? HD 90

? R

? HD ?

45

## RC

RC (stands for READ CHARACTER) reads a single character from the keyboard. Program execution pauses until you type a character. (Also see RS.)

To use the character that RC returns, assign the character to a variable. Then, use the value of the variable in your program.

1. The sample program, COPY, waits for you to type a character. Then, COPY prints the character that you typed. Notice how THROW is used to return you to the Logo command level when you type a zero. This technique is an alternative to using `CTL-G` to halt the program.

Why do you need an alternative to `CTL-G`? When you press `CTL-G` to halt the program, `CTL-G` may be read as a character instead of a control command. You can use `CTL-G` to halt the program, but you may have to enter `CTL-G` several times to catch Logo's attention.

### TO COPY

```
MAKE "C RC
:C = 'Ø'
IFTRUE [ THROW "ERROR [ PRINTNL 'END' ] ]
PRINTNL :C
COPY
END
```

2. When you use the RC command to read the keyboard, you can use certain keys for control purposes. The following program uses COND to define a group of conditional statements. These statements allow you to use **lowercase** E, X, S, and D as control keys (requires 4K/16K RAM cartridge).

Press the **E** key to move the turtle up, press the **X** key to move the turtle down, press the **S** key to move the turtle left, and press the **D** key to move the turtle right. Notice how THROW is used to return you to the Logo command level when you type a zero.

When you try this program, does the turtle always respond to your control keystrokes? Sometimes you have to press a control key more than once before the program pays attention to you. (Also see RS examples.)

TO MOVE

MAKE "C RC

:C='0' IFTRUE [THROW "ERROR  
[PRINTNL 'END' ] ] ]

COND [[:C='e' N 1]

[ :C='x' S 1]

[ :C='s' W 1]

[ :C='d' E 1]]

MOVE

END

## REP number [ action ... ]

REP (stands for REPEAT) performs the list of commands a designated number of times. For example, REP 4 [ FD 10 RT 90 ] draws a box by repeating the list of commands four times.

## RL

RL (stands for READ LIST) reads a line of up to 45 characters, and returns a list. When the RL command is executed, the program halts until you press `RTN`.

1. When you run the following program, type a word or phrase. Then, press `RTN`. The program prints the word or phrase that you typed. Notice that the line is echoed in list format (that is, the character string is enclosed in brackets).

```
TO READ  
  MAKE "LINE RL  
  PRINTNL :LINE  
  READ  
END
```

2. To run this program, type HOME CS BOX. Then, press `RTN`. BOX waits for you to enter a valid color code from 0 to 15. When you press `RTN`, the program draws a box in the specified color. In this program, you use THROW to return you to the Logo command level when you type an X, either lowercase or uppercase.

---

TO BOX

MAKE "C FIRST RL

:C = FIRST [ X ]

IFTRUE [ THROW

"ERROR [PRINTNL 'END' [ ] ]

INK :C

REP 4 [ FD 10 RT 90 ]

BOX

END

---

*You look for the first word in the list that RL returns. If the word is an x or an X, you return to the Logo command level.*

## RN number

RN (stands for RANDOM) generates a random integer from 0 to one less than the number specified. For example, RN 256 generates numbers in the range 0-255.

Negative numbers are treated as positive, and decimal fractions are rounded up. For example, RN 4.1 generates numbers in the range 0-4 instead of 0-3.

TO RANDOM

PRINT RN 4

RANDOM

END

---

## RS

RS (stands for READ STATUS) reads the status of the keyboard. If Logo is waiting to read a character, RS returns the word T (true). Otherwise, if Logo is not waiting to read a character, RS returns the word NIL (false).

To run the following program, called MOVE, enter HOME CS PEN Ø MOVE. Once the turtle starts moving, the turtle continues moving in the same direction until you change directions by pressing one of the following direction keys:

Key	Heading	Key	Heading
e	North	s	West
x	South	d	East

When an out-of-bounds condition is diagnosed, the turtle's heading is rotated right 30 degrees. Press a direction key to direct the turtle away from the screen boundary.

To terminate the program, press **CTL** - **G**, or warm-start Logo.

```
TO MOVE
  CATCH "ERROR [CONTROL]
  THROW "ERROR [RT 30]
  MOVE
END

TO CONTROL
  FD 1
  PUT
```

```

CONTROL
END

TO PUT
  RS
  IFTRUE [MAKE "CTL RC]
  COND [[ :CTL = 'e' HD 90 ]
        [ :CTL = 's' HD 180 ]
        [ :CTL = 'x' HD 270 ]
        [ :CTL = 'd' HD 0 ] ]
END

```

### RT number

RT (stands for RIGHT TURN) rotates the turtle heading right. The input in degrees is subtracted from the current heading.

When the new heading is less than or equal to zero, 360 is added to the heading until the heading falls in the range 0-360 degrees.

```

? HD 180
? RT 90
? HD ?
90
? RT 180
? HD ?
270

```

---

**RUN [ action ... ]**

RUN executes the commands and/or procedures in the list following the command.

Suppose that you had a group of procedures called EAST, WEST, NORTH, and SOUTH. Each procedure moves the turtle 10 units in the direction indicated. The following RUN command executes each procedure in turn until the list of procedures is exhausted.

? RUN [ NORTH EAST SOUTH WEST ]

---

**S number**

S (stands for SOUTH) moves the turtle a specified number of turtle units straight down without changing the turtle's current heading.

Numbers containing a decimal fraction are truncated to integer values (that is, S 10.6 has the same effect as S 10). Negative numbers make S act like N. For example, S -8 is equivalent to N 8.

---

**SAVE 'filename' [ name ... ]**  
or  
**SAVE 'filename' NAMES**

SAVE is used to save a copy of named procedures and/or variables on cassette under the filename you assign. When you use the function NAMES, all of the procedures and variables in memory are saved under the assigned filename on cassette.

---

After saving procedures and variables, use the CHECK command to ensure that the information has been saved properly. To load procedures and/or variables from cassette back into memory, use the LOAD command.

Note that Logo distinguishes between lowercase and uppercase in filenames. Be sure to keep a log of Logo files that you save. Always enter the filename in your log the same way you entered the filename in the SAVE command. (See Chapter 7 for details.)

SAVE 'F1' [LEAPS]

*Save the procedure LEAPS on cassette under filename F1.*

SAVE 'F2'  
[MOVE LEFT RIGHT]

*Save the list of procedures on cassette under filename F2.*

SAVE 'F3' NAMES

*Save all of the procedures and variables in memory on cassette under filename F3.*

SAVE 'F4'  
[ PX PY LEAPS ]

*Save variables PX and PY, and procedure LEAPS on cassette under filename F4.*

## SCAN object

SCAN turns an object into a list. The object may be a number, name, or string. The result is identical to what RL would return if the characters had been typed from the keyboard, except that SCAN returns the word NIL when the object is a list.

? SCAN FIRST POS ?

[ 40 ]

? SCAN 'abc'

[ ABC ]

? SCAN 'A

[ A ]

? SCAN [ A ]

NIL

*The object may not be a list.*

TO COPY

MAKE 'C SCAN RC

PRINTNL :C

COPY

END

*When RC is used with SCAN, each character that you type is turned into a list.*

**SE object ...**

SE (stands for SENTENCE) forms a single list by merging the objects following the command. Objects can be numbers, strings, words, variables, or lists. Enclose the SE command and the list of objects in parentheses when there are more than two objects, or when another command references items in the generated list.

? SE [A B] [C D]	? MAKE 'A [LOGO]
[ A B C D ]	[ LOGO ]
	? MAKE 'B [IS]
	[ IS ]
? SE 'AB' 'CD'	? MAKE 'C [FUN.]
[ 'AB' 'CD' ]	[ FUN. ]
	? (SE :A :B :C)
? (SE [AB] 'C 'D')	[ LOGO IS FUN. ]
[ AB C 'D' ]	

**SHAPE number  
or  
SHAPE ?**

SHAPE changes the appearance of the current turtle when a value from 0 to 255 follows the command. A question mark instead of a number returns the character code for the current turtle's shape. Use SHAPE 255 to change the turtle back to the default shape.

```
? SHAPE 212
? SHAPE ?
212
? SHAPE 255
```

---

## SQRT number

SQRT returns the square root of the specified number. Logo does not calculate the square root of negative numbers.

? SQRT 49

7

---

## SS

SS (stands for SPLIT SCREEN) splits the screen into two regions: the Graphics Screen for turtle graphics, and the 4-line Command Window. Use the SS command to return to the split-screen mode from the full-screen mode (see FS command). SS is the default when you turn Logo on.

---

## STRING word or string ...

STRING returns a single string by appending the words or strings following the command. Enclose the entire expression in parentheses when more than two objects are to be included in the string.

? STRING 'A' 'B'  
'AB'

? (STRING 'L' 'o' 'g' 'o')  
'Logo'

---

## TF

TF returns a list of all the turtles currently in the turtle family.

```
? TF
[ STUDS ANDY FRED JAN ]
```

## THING "name

THING returns the current value of a given variable.

```
? MAKE "PX 40
40
? THING "PX
40
```

```
? THING :PX
SOMEONE DOESN'T LIKE
40
```

*Use a quotation mark instead of dots.*

```
? THING "STUDS
STUDS
```

*When the object is a turtle name, THING returns the turtle's name.*

```
? MAKE "L [LOGO IS FUN.]
[LOGO IS FUN]
? THING "L
[ LOGO IS FUN. ]
```

## THROW "name [command ... procname ]

THROW can be used by itself or in conjunction with CATCH to perform transfers of control.

THROW takes two inputs: a name and a list of commands or procedures to run. The name identifies correspond-

ing CATCH statements. More than one THROW statement may be defined for each CATCH.

Control is normally transferred to THROW from CATCH. Commands in the CATCH list are executed — until a THROW statement with the same label is encountered. Then, the commands in the THROW statement are executed.

Each THROW statement may transfer control to one or more procedures, and multiple THROW statements having the same label may be executed. Control eventually returns to the statement following CATCH. THROW statements may not be executed until their corresponding CATCH is executed (refer to description of CATCH).

Logo issues a THROW "ERROR command internally whenever it encounters errors, and returns you to Logo's command level. Similarly, you can use THROW "ERROR to terminate your program. When "ERROR is not used as a CATCH label, THROW "ERROR transfers control back to the Logo command mode. In effect, THROW "ERROR acts like an END statement in BASIC.

The following program uses THROW to print a message and return you to Logo's command level when the turtle reaches row 1. To run this program, home the turtle, clear the screen, and execute PUT.

```
TO PUT
  FD 1
  (LAST POS ?) = 1
  IFTRUE [THROW "ERROR [PRINTNL
    'STOP. Next step out of bounds' NIL ] ]
  PUT
END
```

Normally, THROW statements may not be executed before corresponding CATCH statements. THROW "ERROR is the only exception. Also, notice how the word NIL is used in THROW to prevent the word NOTHING from being printed after your error message. The word NIL can also be represented by a pair of brackets. For example:

```
IFTRUE [THROW "ERROR [PRINTNL
'STOP. Next step out of bounds' [ ] ] ]
```

In this example, if the program contained a CATCH "ERROR statement, control would return to the statement following CATCH after the message was printed (refer to CATCH examples).

**TILE number**  
or  
**TILE ?**

TILE causes the turtle track to be drawn in the indicated character. Character codes range from 0 to 255. A question mark instead of a number returns the numeric code of the current turtle track character. Use TILE 255 to reset the turtle track to the default character.

? TILE 212

? TILE ?

212

**TN :turtle**

TN returns the HATCH name of the indicated turtle.

? MAKE "X HATCH "FRED

FRED

? TN :X

FRED

**TO procname**  
or  
**TO procname :input ...**

TO allows you to define a new procedure. Enter input variables, if any, following the procedure name.

After you press **RTN** at the end of the TO line, Logo displays a **->** prompt at the beginning of each line.

Then, type the body of the procedure on the first **->** line following the input variables. For example:

```
TO procedure :input ...
:input
-> :input
->
-> procedure body ...
->
-> END
```

*Enter input variables here. Type additional inputs on **->** lines.*

When you type a character in the last position of the text window, Logo automatically advances the character and cursor to the beginning of the next line. Lines that are continued in this manner may contain up to 45 characters. The **->** prompt is not displayed at the beginning of continued lines.

When you reach the end of a continued line, input is inhibited, and the buzzer rings. Press **RTN** to begin a new **->** line. The character in the cursor position at the end of the continued line is discarded. The TO line may be continued the same way **->** lines are continued.

**NOTE:** Do not use **RTN** to divide numbers, words, or strings. If the object does not fit in the remainder of the current line, move the object to the beginning of the next line.

To terminate the procedure definition mode and save the new procedure definition in your workspace, type the keyword **END** at the beginning of a **->** line, and press **RTN**. Logo prints the procedure name on the following line. For example:

? TO HOUSE

-> .

-> .

-> .

->END

HOUSE

After ending the procedure definition, use the **EDIT** command to add, change, or delete lines in the procedure.

## EDITING TEXT

Use the left-arrow key or **CTL**-**H** to delete the character to the left of the cursor. The left arrow key or **CTL**-**H** may only be used on the current line. Continued lines are treated as separate lines.

## USING DELIMITER CHARACTERS

The leading quotation mark at the beginning of a word and the leading apostrophe at the beginning of a string must begin on the same line as the word or string.

In contrast, parentheses and brackets used as delimiters in lists need not appear on the same line as objects in the list.

For example, the following commands are equivalent:

POS ( LIST RN 80 RN 60 )	POS
	(
	LIST
	RN 80
	RN 60
	)

When you end a procedure definition, Logo inserts trailing apostrophe, bracket, and parenthesis delimiter characters in the procedure to match unpaired delimiters. Each bracket or parenthesis that Logo inserts is displayed following the END line. Print or edit procedures in which Logo inserts delimiter characters to ensure that Logo has placed the delimiters where you would have placed them.

End of string can be an apostrophe or end of line.

## TP object

TP (stands for TURTLE PRINT) prints an object on the Graphics Screen without affecting the turtle properties, including the turtle's current position. The object may be a number, string, or word. Use TURTLE Ø to hide the turtle before you use TP. Otherwise, the turtle will cover the first character in the object. Use the INK command before TP to select the color in which the object is displayed.

? TURTLE Ø

? INK 1

? TP 'Logo is fun.'

## TYPE object

TYPE returns a number code that indicates what type of object is identified in the command.

Object	Number Returned
Ø	List
1	Floating-point number
3	Word
9	Character
11	String
14	Turtle
18	Integer

? MAKE "X 1/3

.333333

? TYPE :X

1

? MAKE "X 'Logo is fun.'

'Logo is fun.'  
 ? TYPE :X  
 11

**UNDER number  
 or  
 UNDER ?**

UNDER sets the tile shape under the current turtle, or returns the character code of the tile shape under the current turtle.

**W number**

W (stands for WEST) moves the turtle a specified number of turtle units to the left without changing the turtle's current heading.

Numbers containing a decimal fraction are truncated to integer values (that is, W 10.6 has the same effect as W 10). Negative numbers make W act like E. For example, W -8 is equivalent to E 8.

## WORD object ...

WORD returns a single word by appending the objects following the command. Objects may be words, strings, or numbers. The entire expression must be enclosed in parentheses when more than two objects are identified, and when the expression is used as the object of a Logo command.

```
? (WORD '1 'o 'g 'o)
LOGO
```

## APPENDIX C

### RESERVED WORDS

The following names may be used as variable names, but not procedure names. For example, you can create a variable called FD without affecting the use of the Logo command FD:

```
? MAKE 'FD 1
? 1
```

When you use a reserved word as a procedure name in the TO command, Logo responds with the following error message, and terminates the procedure definition mode:

```
? TO FD
```

```
I DON'T UNDERSTAND 4
```

Similarly, when you create a new procedure in the edit mode, but inadvertently use a reserved word as the procedure name,

Logo displays the message I DON'T UNDERSTAND 4 after you enter the Q command to exit the edit mode normally. To correct the error, simply change the procedure name to a name that is not a reserved word or the name of an existing procedure. Then, reenter the Q command to exit the edit mode. Logo saves the procedure definition in your workspace under the new name.

ALL	FIRST	MEMBER	RUN
ANY	FPUT	N	S
ASK	FREE	NAME?	SAVE
B	FS	NAMES	SCAN
BF	HATCH	NL	SE
BK	HD	NTH	SHAPE
BL	HOME	P	SQRT
CATCH	IFFALSE	PAPER	SS
CH	IFTRUE	PEN	STRING
CHECK	INK	POS	TF
COND	INT	PRINT	THING
CS	L	PRINTNL	THROW
DELAY	LAST	PRINTSE	TILE
DOT	LENGTH	R	TN
E	LIST	RC	TO
EDIT	LOAD	REP	TP
END	LP	RL	TYPE
ER	LPUT	RN	UNDER
F	LT	RS	W
FACE	MAKE	RT	WORD
FD			

---

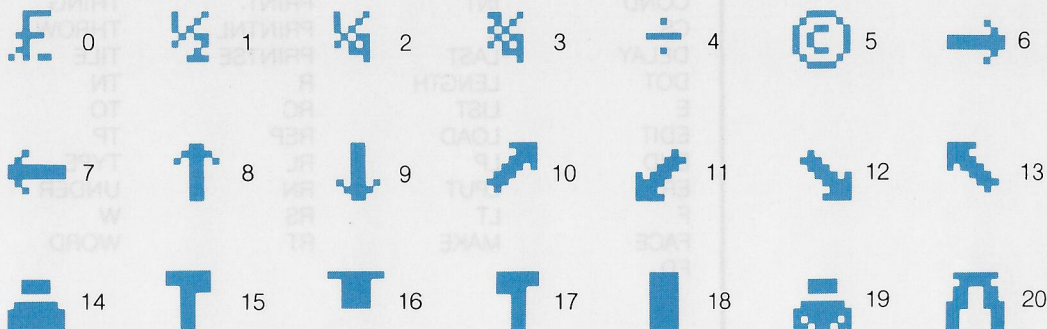
## APPENDIX D

---

### AQUARIUS LOGO CHARACTER SET








Logo uses the Aquarius set of 256 characters. Each character in the Aquarius character set is identified by a number from 0 to 255. Some Logo commands, like TILE and SHAPE, use character codes to change the appearance of the turtle or the turtle tracks.

STUDS, the default Turtle, is represented by character shape 255, an arrow. His trail is TILE 255. To change his SHAPE or TILE, select any of the characters identified below. Type SHAPE number or TILE number. To return to the original shape and trail use SHAPE 255 and TILE 255.



---

	21		22		23		24		25		26		27
---	----	---	----	---	----	---	----	---	----	---	----	---	----

	28		29		30		31		32		33		34
---	----	---	----	---	----	---	----	---	----	---	----	---	----

#	35	\$	36	%	37	&	38	'	39	(	40	)	41
---	----	----	----	---	----	---	----	---	----	---	----	---	----

*	42	+	43	,	44	-	45	.	46	/	47	0	48
---	----	---	----	---	----	---	----	---	----	---	----	---	----

1	49	2	50	3	51	4	52	5	53	6	54	7	55
---	----	---	----	---	----	---	----	---	----	---	----	---	----

8	56	9	57	:	58	;	59	<	60	=	61	>	62
---	----	---	----	---	----	---	----	---	----	---	----	---	----

?	63	@	64	A	65	B	66	C	67	D	68	E	69
---	----	---	----	---	----	---	----	---	----	---	----	---	----

---

---

F	70	G	71	H	72	I	73	J	74	K	75	L	76
M	77	N	78	O	79	P	80	Q	81	R	82	S	83
T	84	U	85	V	86	W	87	X	88	Y	89	Z	90
[	91	\	92	]	93	^	94	_	95	'	96	a	97
b	98	c	99	d	100	e	101	f	102	g	103	h	104
i	105	j	106	k	107	l	108	m	109	n	110	o	111
p	112	q	113	r	114	s	115	t	116	u	117	v	118

---

---

W 119 X 120 Y 121 Z 122 { 123 | 124 } 125

126 127 128 | 129 130 131 132

133 134 135 136 137 138 139

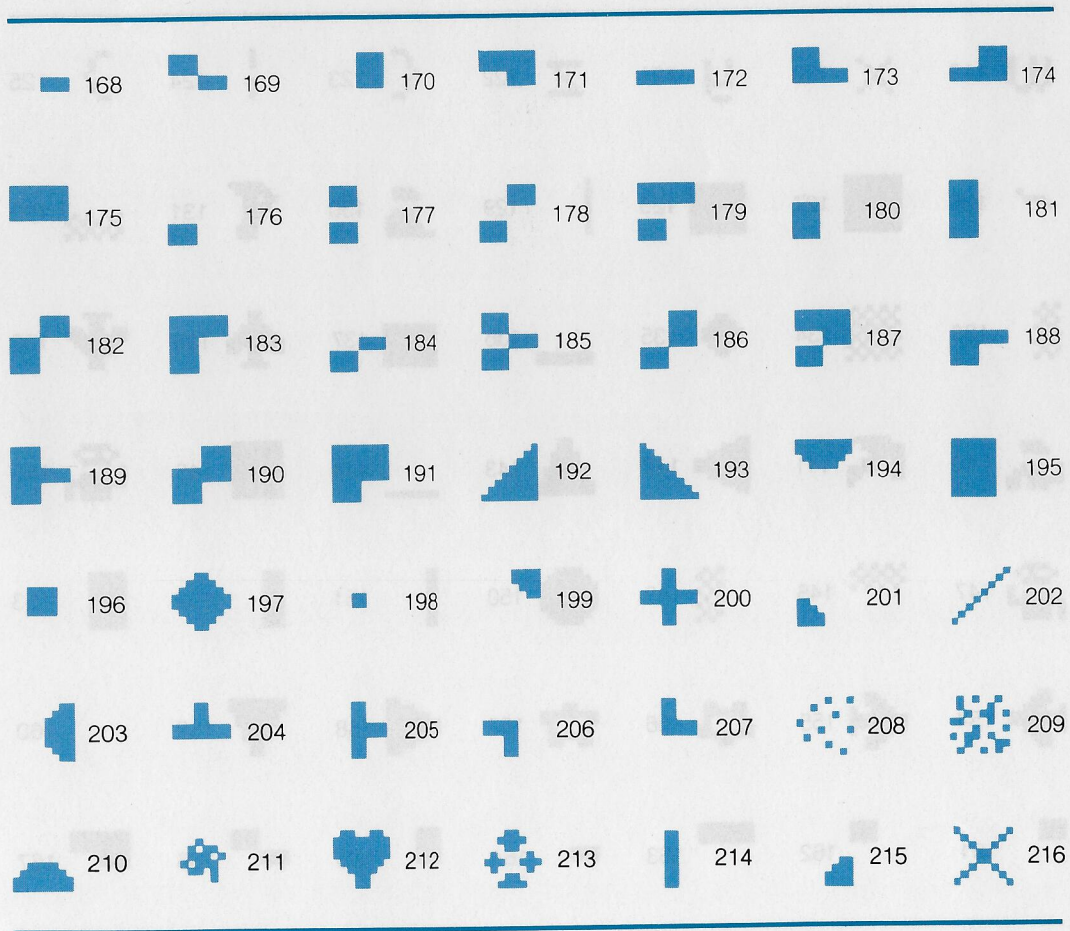
140 141 142 143 144 145 146

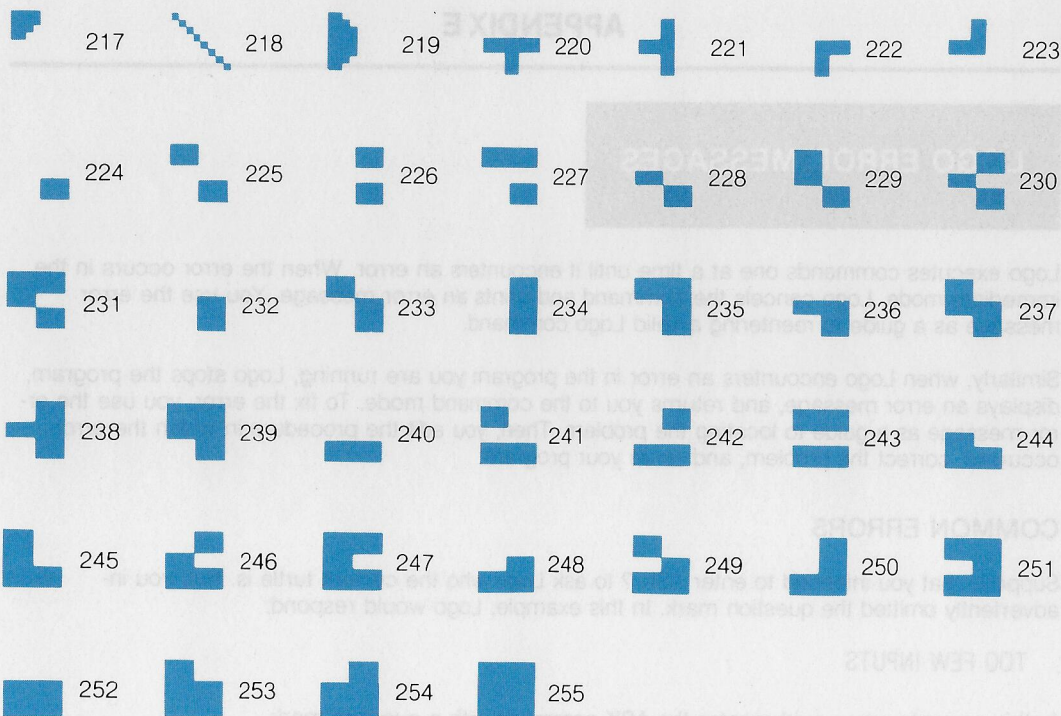
147 148 149 150 151 152 153

154 155 156 157 158 159 160

161 162 163 164 165 166 167

---





---

## APPENDIX E

---

### LOGO ERROR MESSAGES

Logo executes commands one at a time until it encounters an error. When the error occurs in the immediate mode, Logo cancels the command and prints an error message. You use the error message as a guide to reentering a valid Logo command.

Similarly, when Logo encounters an error in the program you are running, Logo stops the program, displays an error message, and returns you to the command mode. To fix the error, you use the error message as a guide to locating the problem. Then, you edit the procedure in which the error occurred, correct the problem, and rerun your program.

### COMMON ERRORS

Suppose that you intended to enter `ASK ?` to ask Logo who the current turtle is. But, you inadvertently omitted the question mark. In this example, Logo would respond:

#### TOO FEW INPUTS

In this example, you would reenter the `ASK` command with a question mark.

A different type of error is indicated when you do not enter spaces between the command name and the command inputs. Spaces between command names and inputs are important.

---

---

When Logo sees `ASK?` without spaces, Logo thinks that you are trying to execute a procedure called `"ASK?"`. If procedure `ASK?` is defined in your workspace, Logo executes the procedure. If procedure `ASK?` is not defined in your workspace, Logo prints the message

### I DON'T KNOW HOW TO ASK?

This type of error is particularly frustrating to first-time Logo users. *You* know how you meant to enter the command. But, when you look at the command, you read the command the way you should have entered it — instead of the way you actually entered it.

## ERRORS IN PROGRAMS

When Logo encounters an error in a program, Logo prints the phrase `NEAR ...` to identify the procedure in which the error occurred. For example, if Logo encountered the command `ASK?` in procedure `HOUSE`, Logo would terminate the program and print the message

### I DON'T KNOW HOW TO ASK? NEAR HOUSE

In this example, you would edit procedure `HOUSE` to correct the `ASK` command, and then rerun the program.

Logo says that the error occurred `NEAR HOUSE` because Logo only knows which procedure is currently being executed. When you are using nesting procedures, the problem being diagnosed may actually occur in a procedure that was called earlier.

---

---

## LOGIC ERRORS

There is one type of error for which Logo does not have an error message. This type of error is called a “logic error.”

A logic error refers to an error in the way you constructed your program. The program runs without generating error messages, but does not give you the results you expect.

Are you calling procedures in the wrong order or using one variable when you should be using another? Only you can tell.

## DEBUGGING

The process of revising and testing a program is called “debugging.” Debugging corrects logic errors, and other types of errors that Logo looks for when you run your program.

## GUIDE TO ERROR MESSAGES

Use the following guidelines to correct errors that Logo diagnoses in commands and programs. Errors that cannot be explained in a short phrase like OUT OF BOUNDS are identified by error message number in the message I DON'T UNDERSTAND ...

Message	Description
I DON'T KNOW HOW TO ...	Indicates that the identified command name or procedure name does not exist. Correct the command name or procedure name, and try again.  For example, if you meant to enter PAPER 10 — but entered PAPE 10 or PAPER10 instead — Logo responds I DON'T KNOW HOW TO PAPE

Message	Description
OUT OF BOUNDS	<p>or I DON'T KNOW HOW TO PAPER1Ø.</p> <p>Before you move the turtle, Logo checks whether the destination would place the turtle beyond the graphic screen boundaries. In the immediate mode, reenter the command with a smaller value. In a procedure, use Logo commands to determine whether moving the turtle would place the turtle beyond the graphic screen boundaries <b>before</b> the turtle is actually moved.</p>
SOMEONE DOESN'T LIKE ...	<p>Indicates that Logo encountered a syntax error in a command.</p> <p>For example, FD [ 1Ø ] terminates in the message SOMEONE DOESN'T LIKE [ 1Ø ]. In this example, Logo objects to the use of brackets: the FD command does not use list input.</p>
TOO FEW INPUTS	<p>When Logo objects to an empty list, Logo uses the word NIL. For example, BF [] terminates in the message SOMEONE DOESN'T LIKE NIL.</p> <p>You omitted one or more inputs to a command or procedure that uses inputs. For example, INK expects an ink value. If you enter INK without a valid ink value, Logo terminates the command with the message TOO FEW INPUTS.</p>
NO ROOM	<p>The NO ROOM message indicates that there is insufficient workspace to continue. The NO ROOM condition may be encountered in the command mode or edit mode, or during program execution. (See "Coping with NO ROOM" at the beginning of this reference section.)</p> <p>Workspace is measured in bytes. A "byte" is the smallest unit of</p>

---

## Message

## Description

memory in which a character such as a digit or letter of the alphabet can be stored.

The FACE command — and commands that return a floating-point result — require 256 bytes of workspace to do the arithmetic. The NO ROOM message indicates that the required amount of workspace is not available. To continue, erase unnecessary procedures, or save existing procedures on cassette, and use a larger RAM cartridge.

When you use the TO command to create a procedure, the procedure definition is terminated with a NO ROOM message when the remaining workspace is insufficient to complete the procedure; the current procedure definition is deleted. Erase unnecessary procedures, or save existing procedures on cassette, and use a larger RAM cartridge.

When you use the Q command to exit the edit mode, Logo deletes the original version of the procedure before it stores the new version in your workspace. If Logo displays the NO ROOM message and does not allow you to exit, try the Q command a second time. After deleting the old version of the procedure, Logo may find enough space to save the new version. If the NO ROOM message is still displayed in the edit mode, you cannot save the procedure. Warm start Logo to return to the command mode (the procedure definition is deleted). Erase unnecessary procedures and redefine the procedure, or save the existing procedures on cassette and use a larger RAM cartridge.

I DON'T  
UNDERSTAND ...

The number that follows the message indicates the specific error.

Message	Description
1	Incomplete list. Look for a missing bracket, parenthesis, or apostrophe.
4	Cannot change Logo reserved word. You inadvertently used an Aquarius Logo reserved word in a TO or EDIT command. If error 4 is diagnosed when you attempt to exit the edit mode, change the procedure name, and enter the Q command again to exit.
5	<p>A value of NOTHING was used as an input to a Logo command that does not use NOTHING as an input. For example, when you enter PRINT FD 1Ø, Logo returns NOTHING:</p> <pre>? PRINT FD 1Ø NOTHING</pre> <p>Logo prints the word NOTHING, because FD 1Ø — as a function — does not return a value. Similarly, when FD 1Ø is used as an input to a Logo command like FD, Logo returns I DON'T UNDERSTAND 5. In this example, the FD command expects a numeric input. The command cannot use NOTHING as a value.</p> <pre>? FD FD 1Ø I DON'T UNDERSTAND 5</pre>
13	Word or string too long. When you define or edit procedures, you do not use RTN to divide words or strings. Consequently, the length of a word or string is dictated by the line length. In list processing, however, you can form longer words or strings. When you use a 16K RAM cartridge, words or strings that you form through list processing may be up to 89 characters in length. When you use less RAM, the maximum

Message	Description
	length of words and strings is less (see "Words and Strings" at the beginning of this reference section).
26	A THROW command was executed before a corresponding CATCH command was executed.
33	The procedure is too large to edit. Erase the procedure, and redefine the procedure as a group of smaller procedures. Define procedures in the edit mode whenever you are unsure whether a given procedure will fit on the edit screen.
34	Mismatched brackets and/or parentheses.
37	Input out of range. A number of Logo commands expect inputs to be within a certain range of values, for example 0-15. INK 20 terminates in error 37.
38	Incomplete string: an apostrophe is missing.
40	Procedure mismatch. Logo encountered a mismatch while using CHECK to verify procedures in a cassette file against corresponding procedures in your workspace. Use CHECK after saving or loading procedures. If error 40 occurred after saving procedures, resave the procedures, and enter CHECK again. If error 40 occurred after loading procedures, reload the procedures, and enter CHECK again.
110	The number is too large or too small to be represented in Aquarius Logo's range of numbers (that is, larger than $10^{38}$ or smaller than $10^{-38}$ ).

Message	Description
120	Illegal division by zero.
255	<p>The error was so confusing that Logo did not know where or how to continue. Depending on the severity of the problem, you may lose the procedures in your workspace. If Logo does not respond when you press <b>RTN</b>, try warm starting Logo. If a warm start does not bring you back to the command mode, turn the console off, wait a few seconds, and then turn the console back on.</p>

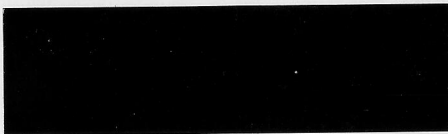
---

## 90 DAY LIMITED WARRANTY

Mattel Electronics warrants to the original consumer purchaser of any AQUARIUS™ cartridge, cassette or disk it manufactures, that the product will be free of defects in material or workmanship for 90 days from the date of purchase under normal in-home use.

Mattel Electronics will not assume any liability or responsibility for loss or damage, direct or indirect, caused by or alleged to be caused by any Aquarius software programs or the use made of any such program by the consumer. This disclaimer includes but is not limited to any interruption of service, loss of money, or anticipatory profits resulting from the use or operation of such programs.

Mattel Electronics sole obligation under this warranty will be to repair or replace the defective product, at its option. If defective, return the Aquarius cartridge, cassette or disk along with proof of the date-of-purchase to either your local dealer or postage prepaid to:



Mattel Electronics Service Center (West)  
13040 East Temple Avenue  
City of Industry, California 91746

---

This warranty excludes incidental or consequential damages resulting from the product or use of the product. (Some states do not allow the exclusion of incidental or consequential damages, so the above exclusion may not apply to you.)

This warranty gives you specific legal rights and you may also have other rights which vary from state to state. This warranty does not cover damage resulting from purchaser abuse, accident, negligence, or damages subsequent to purchase.

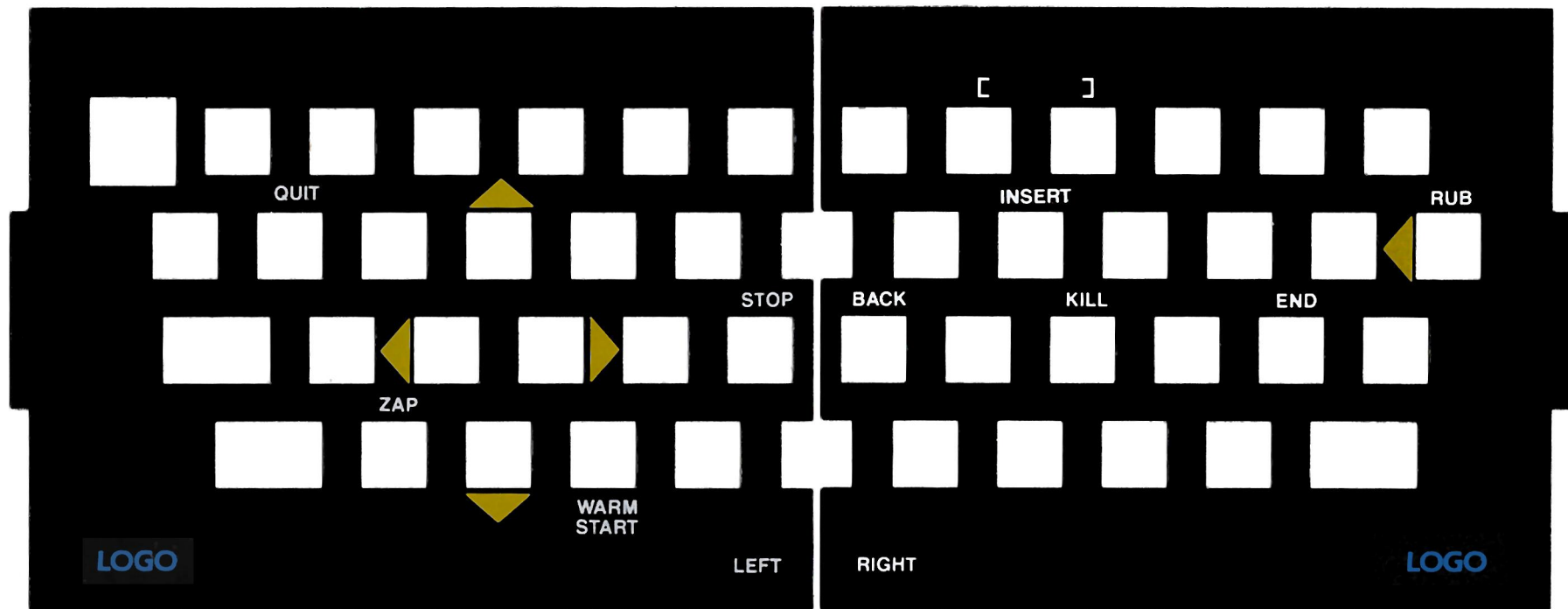
---

4391-0920



**MATTEL ELECTRONICS®**

Illustrations: © Mattel Electronics, Inc. 1983. Hawthorne, CA 90250.  
**PRINTED IN HONG KONG.** All Rights Reserved.



A black rectangular box, possibly a tape or a small case, with a blue 'LOGO' label on the left side. The box has a horizontal band across the top and a recessed rectangular area in the center where the label is located. The background is a light gray surface.

**LOGO**

A black, rectangular object with a series of horizontal ridges or grooves across its upper portion. Below this textured area is a smooth black band containing the word "LOGO" in white capital letters. The object has a slightly raised base at the bottom.

**LOGO**

**AQUARIUS**<sup>™</sup>  
HOME COMPUTER SYSTEM

LOGO

0



MATTEL ELECTRONICS

**AQUARIUS**  
HOME COMPUTER SYSTEM

LOGO

No. 4391

MATTEL ELECTRONICS



LOGO

AQUARIUS  
HOME COMPUTER SYSTEM

4391-0910

AQUARIUS  
HOME COMPUTER SYSTEM

LOGO

The revolutionary computer language!

- Quickly teaches you or your child computer programming!
- "Turtle graphics" make the learning fun!

Enclosed cartridge for use with  
AQUARIUS Home Computer System.  
MATTTEL ELECTRONICS®



AQUARIUS  
HOME COMPUTER SYSTEM

LOGO

No. 4391

MATTTEL ELECTRONICS

AQUARIUS  
HOME COMPUTER SYSTEM

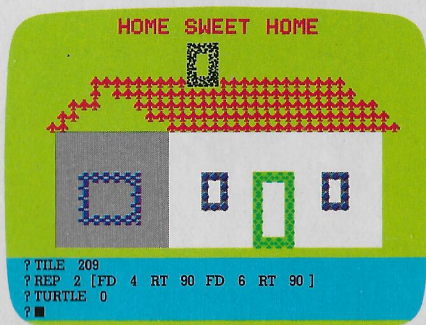
# AQUARIUS<sup>TM</sup> HOME COMPUTER SYSTEM LOGO

No. 4391

This package contains one Aquarius LOGO cartridge,  
two keyboard overlays and instruction manual.

## LOGO cartridge

Designed specially to introduce you & your child to computers & programming! It's educational and it's fun!  
Learn simple commands and create graphics on your TV screen. See results step-by-step! LOGO starts out  
simple...and gets as complex as you want! It makes learning computer programming fun!



- ☐ Type simple commands...see graphics immediately on the TV screen. Correct errors as you go. Even create your own commands!
- ☐ Learn problem solving and analytical skills as you design your own graphics and games!
- ☐ Use LOGO to help with many arithmetic operations...including geometry!
- ☐ Learn advanced programming. Compose, edit and manipulate your own writing with "list processing"!

Create graphics while you learn to program!

Can be used with the AQUARIUS Computer alone or  
with the Mini Expander. Requires a cassette recorder  
for permanent memory storage. Each sold separately.  
Extensive list processing requires memory cartridge.

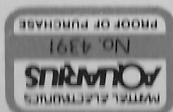
**MATTEL ELECTRONICS<sup>®</sup>**

© Mattel Electronics, Inc. 1983. Hawthorne, CA 90250. MADE IN SINGAPORE.  
OVERLAYS PRINTED IN HONG KONG. Manufactured for Mattel. All Rights  
Reserved. ® and TM designate U.S. trademarks of Mattel, Inc. (L001)

MATTEL ELECTRONICS

# LOGO

AQUARIUS<sup>TM</sup>  
HOME COMPUTER SYSTEM



4391-0910